Oracle® Application Server InterConnect

Adapter for HTTP Installation and User’s Guide

10*g* (9.0.4)

Part No. B10413-01

September 2003

Oracle Application Server InterConnect Adapter for HTTP Installation and User's Guide, 10*g* (9.0.4)

Part No. B10413-01

Contributing Authors: Ching Chung, Mark Kennedy, and Min Yao

Contents

Send Us Your Comments

Oracle Application Server InterConnect Adapter for HTTP Installation and User's Guide, 10*g* (9.0.4)

Part No. B10413-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, 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 (optionally) electronic mail address.

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

Preface

This guide is the primary source of introduction, installation, configuration, and usage information for the Hypertext Transfer Protocol (HTTP) adapter.

This preface contains these topics:

Audience

Oracle Application Server InterConnect Adapter for HTTP Installation and User's Guide is intended for developers who want to integrate an HTTP application with other applications using Oracle Application Server 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 Corporation 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Organization

This document contains:

Chapter 1, "Introduction"

This chapter describes the Oracle Application Server 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 for the HTTP adapter.

Chapter 4, "Frequently Asked Questions"

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

Appendix A, "adapter.ini Example File"

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

Related Documentation

For more information, see these Oracle resources:

- *Oracle Application Server InterConnect User's Guide*
- *Oracle Application Server InterConnect Installation Guide*
- *Oracle Application Server InterConnect Adapter Configuration Editor User's 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://otn.oracle.com/membership/`

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

`http://otn.oracle.com/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 Code Examples
- Conventions for Microsoft Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and 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.	*Oracle9i Database 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, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, 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.

Convention	Meaning	Example
lowercase monospace (fixed-width) font	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter sqlplus to open 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. Set the QUERY_REWRITE_ENABLED initialization parameter to true. 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 U*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
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	DECIMAL (*digits* [, *precision*])
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]

Convention	Meaning	Example

`. . .`	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		`CREATE TABLE ... AS` *`subquery`*`;`
		
		`SELECT` *`col1`*`,` *`col2`*`, ... ,` *`coln`* `FROM employees;`

. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points 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. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates 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 Microsoft Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Oracle Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > 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 file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`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>`
	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 scott/tiger TABLES=emp QUERY=\"WHERE job=’SALESMAN’ and sal<1600\"` `C:\>imp SYSTEM/`*`password`* `FROMUSER=scott TABLES=(emp, dept)`
`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`

1 Introduction

This chapter describes the Hypertext Transfer Protocol (HTTP) adapter and the hardware and software requirements.

This chapter contains these topics:

- HTTP Adapter Overview
- HTTP Adapter System Requirements
- HTTP Adapter Known Limitations

Note: 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).

HTTP Adapter Overview

The HTTP adapter enables an HTTP application to be integrated with other applications using Oracle Application Server InterConnect (OracleAS InterConnect). This adapter is useful in all enterprise application integration (EAI) environments that use the HTTP transport protocol. EAI is the integration of applications and business processes within the same company (known as an enterprise).

The HTTP adapter performs the following tasks:

- Monitors incoming messages received in the form of HTTP requests to the HTTP adapter servlet
- Sends messages to remote Web servers through a proxy host

The payload type (the data being delivered to a destination) for the HTTP adapter is one of the following:

- Extensible Markup Language (XML) data
- Data definition description language (D3L) data

The HTTP adapter supports HTTP versions 1.0 and 1.1. The HTTP adapter also provides SSL functionality (known as HTTPS) with Oracle JavaSSL, which uses a wallet generated from Oracle Wallet Manager. You can also select cipher suites suitable for an SSL connection. Cipher suites control the combination of encryption and data integrity used by SSL. The HTTP adapter also supports sending HTTP requests and receiving HTTP replies through a proxy server.

Figure 1–1 depicts the data flow of incoming messages (from an HTTP client to OracleAS InterConnect). Incoming messages are sent to a servlet, provided by the HTTP adapter, that relays the message to an HTTP receiver in the adapter through a remote method invocation (RMI).

Figure 1–1 Incoming Messages

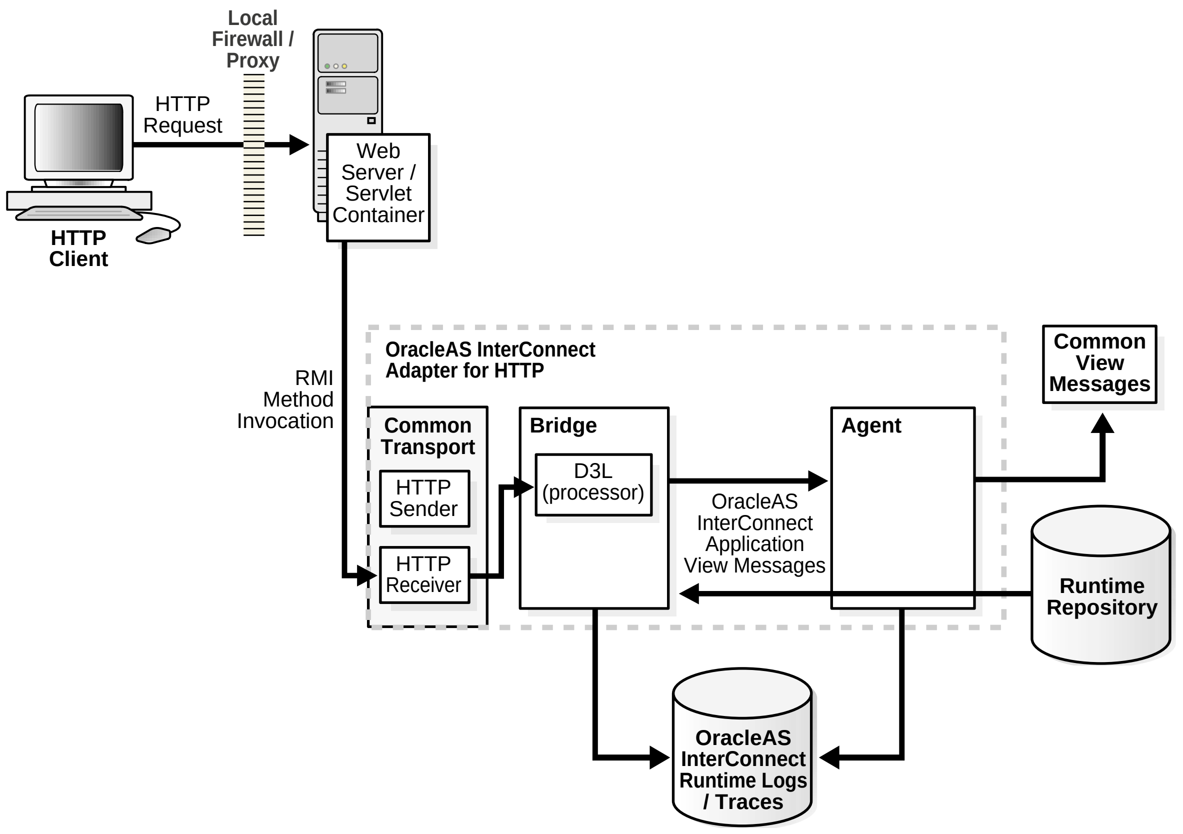
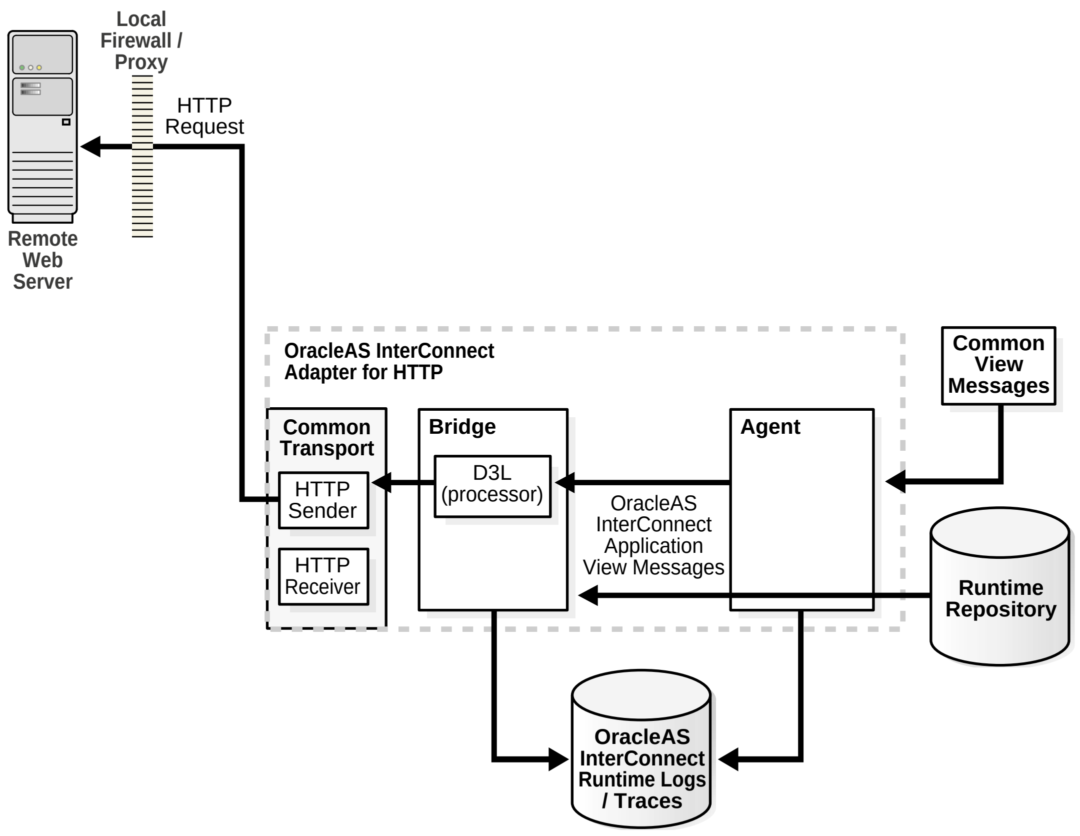


Figure 1–2 depicts the data flow of outgoing messages (from OracleAS InterConnect to a remote Web server). Outgoing messages are sent from an HTTP sender object in the adapter to a remote Web server.

Figure 1–2 Outgoing Messages



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

HTTP Adapter System Requirements

The following sections describe HTTP adapter system requirements:

- Hardware Requirements
- Software Requirements

Hardware Requirements

Table 1–1 lists the hardware requirements for the computer on which the Oracle HTTP adapter is to be installed.

Table 1–1 Hardware Requirements

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

Software Requirements

The following sections describe HTTP adapter software requirements:

- Operating System Requirements
- JRE Requirements
- Servlet Requirements

Operating System Requirements

Table 1–2 lists the operating system requirements for the computer on which the HTTP adapter is to be installed.

Table 1–2 Operating System Requirements

Operating System	Version
Windows NT	Version 4.0 with Service Pack 6 or above
Windows 2000	With Service Pack 1 or above
IBM AIX 5L	5.1 and 5.2 (64 bit)
HP Tru64	5.1a and 5.1b (64 bit)
HP-UX	11.0 and 11.11 (64 bit)

Table 1–2 Operating System Requirements

Operating System	Version
Red Hat Linux	Advanced Server 2.1
Sun SPARC Solaris	8 and 9 (32 bit)

JRE Requirements

OracleAS InterConnect uses Java Runtime Environment (JRE) 1.4.1, which is installed with its components.

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 on which the Oracle HTTP adapter is installed.

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

HTTP Adapter Known Limitations

The HTTP adapter has the following limitations:

- Only the publish / subscribe model is supported.
- All the incoming messages for a HTTP adapter application are received through a single HTTP adapter servlet.
- The sending endpoint and receiving endpoint are restricted to HTTP endpoints.
- Only the HTTP `POST` access method is supported by the HTTP adapter servlet for receiving incoming messages.

2 Installation and Configuration

This chapter describes the installation and configuration of the HTTP adapter in the following topics:

- Installing HTTP Adapter
- HTTP Adapter Configuration Parameters

Installing HTTP Adapter

This section contains these topics:

- Preinstallation Tasks
- Installation Tasks
- Postinstallation Tasks

Preinstallation Tasks

The HTTP adapter must be installed in one of the following Oracle homes:

- An existing OracleAS InterConnect Oracle home for 10*g* (9.0.4)
- A new Oracle home (Oracle Universal Installer creates this Oracle home for you)

Consult the following guides before proceeding with HTTP adapter installation:

- *Oracle Application Server Installation Guide*, which includes information on:
  - Oracle Universal Installer startup
- *Oracle Application Server InterConnect Installation Guide*, which includes information on:
  - CD-ROM mounting
  - OracleAS InterConnect software, hardware, and system requirements
  - OracleAS InterConnect installation

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

Installation Tasks

To install the HTTP adapter:

1. On the Available Product Components page of the OracleAS InterConnect installation, select HTTP adapter, then select **Next**.

   Consider the following scenarios:

   - If installing the HTTP adapter in an independent Oracle home, make sure that the OracleAS InterConnect Hub has been installed, not necessarily in the same Oracle home. Continue to step 2.
   - If installing the HTTP adapter in an existing Oracle home, make sure that it is a home directory to one of the OracleAS InterConnect component. Continue to step 3.

   Note: The hub database information, such as the SID, host, port, and username/password from the Hub installation, is needed for step 2.

2. If installing OracleAS InterConnect for the first time on this machine, complete the following steps to enter the hub database information:

   a. On the Welcome page, select **Next**. The Database Configuration page displays. Enter information in the following fields:

      * Host Name—The host name of the machine where the hub database is installed.
      * Port Number—The TNS listener port for the hub database.
      * Database SID—The SID for the hub database.

   b. Click **Next**. The Database User Configuration page displays. Enter information in the following fields:

      * User Name—The hub database user name. Make sure the OracleAS InterConnect Hub is installed. If the Hub is not installed, complete the installation and note the user name and password.
      * Password—The password for the hub database user.

3. Click **Next**. The Adapter Configuration page displays. Enter the application to be defined or already defined in iStudio in the Application Name field. White spaces or blank spaces are not permitted. The default value is `myHTTPApp`.

4. Click **Next**.

   The OracleAS InterConnect HTTP Adapter usage screen appears.

5. Select one of the following options and go to the step specified to enable the sending and receiving of messages, the sending of messages only, or the receiving of messages only from an external data source, such as an HTTP server. You can change the values for these selections later by editing parameter settings in the `adapter.ini` file.

If You Select...	Then Click Next and Go to Step...
Configure for both sending and receiving messages	6
Configure for sending messages ONLY	6
Configure for receiving messages ONLY	8

6. Enter the following information in the OracleAS InterConnect HTTP Adapter Configuration - Configure sending endpoint information screen:

   - URL—The URL of the outgoing HTTP server to which OracleAS InterConnect sends messages. Enter the URL as follows:

     `http://`*`hostname`*`:`*`port/path`*

7. Click **Next**.

   The installation screen that appears next is based on the selection you made in Step 5:

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

8. Enter the following information in the OracleAS InterConnect HTTP Adapter Configuration - Configure receiving endpoint information screen:

   - Hostname—The hostname of the HTTP server from which OracleAS InterConnect receives messages.
   - Port Number—The port number of the HTTP server.

9. Click **Next**.

10. Complete any other fields for other components selected for installation, such as other adapters.

    When finished, the Summary screen appears.

11. Select **Install** to install the HTTP adapter. The adapter is installed in the following directory:

Platform	Directory
Windows	*ORACLE_HOME*\oai\9.0.4\adapters\Application
UNIX	*ORACLE_HOME*/oai/9.0.4/adapters/*Application*

Application is the value you specified in Step 3 on page 2-3. A webapps subdirectory is created in the *Application* directory identified in Step 11. webapps includes the following files created 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

Postinstallation Tasks

HTTP adapter installation creates an adapter.ini file that consists of configuration parameters read by the HTTP adapter at startup. The configuration parameter settings are appropriate for most HTTP application environments. If you want to customize some adapter.ini file parameter settings for the HTTP application, see the following sections:

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

See Also:

- Table 2–1 on page 2-11 for the location of the `adapter.ini` file
- Table 2–7 on page 2-14 for `adapter.ini` file parameter setting information specific to the HTTP adapter

Customizing the Payload Datatype

Payload data is the data sent between applications. If you want to change the payload datatype from the default of XML to the data definition description language (D3L), edit the following parameters in the `adapter.ini` file.

To customize the payload datatype:

1. Set the `ota.type` parameter to the payload type `D3L`. For example:

   ```
   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 on page 2-22 for additional information

Customizing the Sending Endpoints

If you want to customize the behavior of the sending endpoints (destinations) for messages, edit the following parameter in the `adapter.ini` file.

To customize the sending endpoints:

1. Set the `http.sender.timeout` parameter to the timeout interval in milliseconds for HTTP connections. This parameter automatically defaults to a value of `60000` during installation. For example:

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

   See Also: `http.sender.timeout` parameter description on page 2-22 for additional information

Customizing the Authentication Scheme

If you want to use an authentication scheme, edit the following 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 username. 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: The following parameter descriptions for additional information:

   - `http.sender.authtype` on page 2-22
   - `http.sender.realm` on page 2-22
   - `http.sender.username` on page 2-22
   - `http.sender.password` on page 2-23
   - "How do I make the adapter.ini file password parameters secure?" on page 4-6 for instructions on encrypting the `http.sender.password` parameter password

Customizing a Proxy Host

If you want to use a proxy host, edit the following parameters in the `adapter.ini` file. These parameters are not automatically 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: The following parameter descriptions for additional information:

   - `http.sender.proxy_host` on page 2-23
   - `http.sender.proxy_port` on page 2-23

Customizing a Secure Socket Layer Environment

If you want to use the secure socket layer (SSL) environment for sending messages, edit the following parameters in the `adapter.ini` file. These parameters are not automatically set to default values during installation.

To customize a secure socket layer 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 some cases, you may need to use Oracle Wallet manager to add additional trusted certificates from the HTTP server into your wallet 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
   ```

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:

- `http.sender.wallet_location` on page 2-23
- `http.sender.wallet_password` on page 2-23
- `http.sender.cipher_suites` on page 2-24
- "How do I make the adapter.ini file password parameters secure?" on page 4-6 for instructions on encrypting the `http.sender.wallet_password` parameter password

Customizing the Receiving Endpoints

If you want to customize the behavior of receiving endpoint messages, edit the following parameter in the `adapter.ini` file.

To customize the receiving endpoints:

1. Set the `http.receiver.registry_port` parameter to the remote method invocation (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: `http.receiver.registry_port` parameter description on page 2-24 for additional information

Manually Deploying an EAR File

If the Oracle Application Server Containers for J2EE (OC4J) is installed on a separate computer from the HTTP adapter, you must manually:

- Edit the `web.xml` file
- Deploy the EAR file (`oai.ear`) located in the directory of the HTTP adapter

To manually deploy an EAR file:

1. Go to the following directory:

   ```
   cd myHTTPapphome
   ```

   where *`myHTTPapphome`* is the directory in which the HTTP application is installed and the value you defined in Step 3 on page 2-3.

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

   ```
   jar xvf oai.ear
   ```

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

   ```
   jar xvf oai.war
   ```

4. Go to the following directory:

   ```
   cd WEB-INF
   ```

5. Use a text editor to open the `web.xml` file:

   `web.xml` specifies the RMI information that must match with the setting in the `adapter.ini` file. For example, `rmiHost` must match the hostname of the computer on which the HTTP adapter is installed. The HTTP adapter serves 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.

   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>
```

6. Save the changes and exit the file.
7. Return to the following directory:

   ```
   cd myHttpApphome
   ```

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

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

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

HTTP Adapter Configuration Parameters

Table 2–2, Table 2–3, and Table 2–4 describe executable files, configuration files, and directories. These files and directories are accessible from the directory shown in Table 2–1:

Table 2–1 HTTP Adapter Directory

Platform	**Directory**
UNIX	*ORACLE_HOME*`/oai/9.0.4/adapters/Application`
Windows	*ORACLE_HOME*`\oai\9.0.4\adapters\Application`

Table 2–2 HTTP Executable Files

File	**Description**
`start.bat` (Windows) `start` (UNIX)	Takes no parameters, starts the adapter
`stop.bat` (Windows) `stop` (UNIX)	Takes no parameters; stops the adapter

Table 2–2 HTTP Executable Files

File	Description
`ignoreerrors.bat` (Windows) `ignoreErrors` (UNIX)	If an argument is specified, then the given error code is ignored: `ignoreerrors` *`errorCodeToBeIgnored`* If no argument is specified, then all error codes specified in the `ErrorCodes.ini` file are ignored: `ignoreerrors`

See Also: "HTTP Adapter Error Codes" on page 3-16

Table 2–3 HTTP Configuration Files

File	Description
`ErrorCodes.ini` (Windows and UNIX)	Contains one error code per line
`adapter.ini` (Windows and UNIX)	Consists of all the initialization parameters that the adapter reads at startup

See Also: Appendix A, "adapter.ini Example File"

Table 2–4 HTTP Directories

Directory	Description
`persistence`	The messages are persisted (made available) in this directory. Do not edit this directory or its files.
`logs`	The logging of adapter activity is done in subdirectories of the `logs` directory. Subdirectory names take the following form: *`timestamp_in_milliseconds`* Each time the adapter is run, a new subdirectory is created in which logging is done in an `oailog.txt` file.

Hub.ini Parameter File

The HTTP adapter connects to the hub database using parameters from the `hub.ini` file located in the hub directory. Table 2–5 lists the parameter name, description, the possible and default values, and example for each parameter

Table 2–5 Hub.ini Parameters

Parameter	Description	Example
`hub_username`	The name of the hub database schema (or username). The default value is `oaihub904`.	`hub_username=oaihub904`
`hub_password`	The password for the hub database user. There is no default value. You input the `hub_password` value during installation.	`hub_password=manager`
`hub_host`	The name of the machine hosting the hub database. There is no default value. You input the `hub_host` value during installation.	`hub_host=mpjoshipc`
`hub_instance`	The valid SID of the hub database. There is no default value. You input the `hub_instances` value during installation.	`hub_instance=orcl`
`hub_port`	The TNS listener port number for the HUB database instance. There is no default value. You input the `hub_port` value during installation.	`hub_port=1521`
`repository_name`	The valid name of the repository this adapter talks to. The default value is `InterConnectRepository`.	`repository_name=InterConnectRepository`

RAC-specific Hub.ini Parameters

When a hub is installed on a Real Application Cluster (RAC) database, parameters listed in Table 2–6 represent information on additional nodes used for connection and configuration. These parameters are added on top of the default parameters which represent the primary node. In Table 2–6, `x` represent the node number, which varies between 2 and the number of nodes. For example, if the RAC setup contains 4 nodes, `x` can take a value between 2 and 4.

Table 2–6 RAC-specific Hub.ini Parameters

Parameter	Description	Example
`hub_num_nodes`	Number of nodes in Real Application Clusters.	`hub_num_nodes=4`
`hub_hostx`	The host where the Real Application Clusters database is installed.	`hub_host2=dsunram13`
`hub_instancex`	The instance on the respective node.	`hub_instance2=orcl2`
`hub_portx`	The port on which the listener is listening.	`hub_port2=1521`

Agent Connection Parameters

The agent component of the HTTP adapter reads the `adapter.ini` file at runtime to access HTTP adapter parameter configuration information. Table 2–7 lists the parameter name, a description for each parameter, the possible and default values, and an example.

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
`application`	Specifies the name of the application to which this adapter connects. This must match with the name specified in iStudio during creation of metadata. Use any alphanumeric string. There is no default value.	`application=aqapp`
`partition`	Specifies the partition this adapter handles as specified in iStudio. Any alphanumeric string is a possible value. There is no default value.	`partition=germany`
`instance_number`	Specifies the instance number to which this adapter corresponds. Specify a value only if you want to have multiple adapter instances for the given application with the given partition. Possible values are any integer greater than or equal to `1`. There is no default value.	`instance_number=1`
`agent_log_level`	Specifies the amount of logging necessary. Possible values are: `0`=errors only `1`=status and errors `2`=trace, status, and errors The default value is `1`.	`agent_log_level=2`
`agent_subscriber_name`	Specifies the subscriber name used when this adapter registers its subscription. The possible value is a valid Oracle Advanced Queue subscriber name. There is no default value.	`agent_subscriber_name=httpapp`
`agent_message_selector`	Specifies conditions for message selection when registering its subscription with the hub. The possible value is a valid Oracle Advanced Queue message selector string. There is no default value.	`agent_message_selector=recipient_list`, like `'%,aqapp,%'`

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
`agent_reply_subscriber_name`	Specifies the subscriber name used when multiple adapter instances for the given application with the given partition are used. This parameter is optional if only one instance is running. The possible value is the application name (*`parameter`*:*`application`*) concatenated with the instance number (*`parameter`*:*`instance_number`*). There is no default value.	If `application=httpapp` and `instance_number=2`, then `agent_reply_subscriber_name=httpapp2`
`agent_reply_message_selector`	Used only if multiple adapter instances exist for the given application with the given partition. The possible value is a string built using the concatenated application name (*`parameter`*`:`*`application`*) with the instance number (*`parameter`*`:`*`instance_number`*). There is no default value.	If `application=httpapp` and `instance_number=2`, then `agent_reply_message_selector=recipient_list`, like `'%,httpapp2,%'`
`agent_tracking_enabled`	Specifies if message tracking is enabled. Set this parameter to `false` to turn off all tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. Possible values are `true` or `false`. The default value is `true`.	`agent_tracking_enabled=true`
`agent_throughput_measurement_enabled`	Specifies if the throughput measurement is enabled. Set this parameter to `true` to turn on all throughput measurements. Possible values are `true` or `false`. The default value is `true`.	`agent_throughput_measurement_enabled=true`
`agent_use_custom_hub_dtd`	Specifies whether to use a custom document type definition (DTD) for the common view message when handing it to the hub (the repository in which metadata is stored). By default, adapters use an OracleAS InterConnect-specific DTD for all messages sent to the hub, as other OracleAS InterConnect adapters retrieve the messages from the hub and know how to interpret them. Set this parameter to `true` if for every message, the DTD imported for the message of the common view is used instead of the OracleAS InterConnect DTD. Only set this parameter to `true` if an OracleAS InterConnect adapter is not receiving the messages from the hub. Possible values are `true` or `false`. There is no default value.	`agent_use_custom_hub_dtd=false`

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
agent_metadata_caching	Specifies the metadata caching algorithm. Possible values are: ■ startup—Cache everything at startup. This may take a while if there are a lot of tables in the repository. ■ demand—Cache metadata as it is used. ■ none—No caching. This slows down performance. The default value is demand.	agent_metadata_caching=demand
agent_dvm_table_caching	Specifies the domain value mapping (DVM) table caching algorithm. Possible values are: ■ startup—Cache all DVM tables at startup. This may take a while if many tables are in the repository. ■ demand—Cache tables as they are used. ■ none—No caching. This slows down performance. The default value is demand.	agent_dvm_table_caching=demand
agent_lookup_table_caching	Specifies the lookup table caching algorithm. Possible values are: ■ startup—Cache all lookup tables at startup. This may take a while if many tables are in the repository. ■ demand—Cache tables as they are used. ■ none—No caching. This slows down performance. The default value is demand.	agent_lookup_table_caching=demand
agent_delete_file_cache_at_startup	With any of the agent caching methods enabled, metadata from the repository is cached locally on the file system. Set this parameter to true to delete all cached metadata on startup. Possible values are true or false. The default value is false. **Note:** After changing metadata or DVM tables for this 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_max_ao_cache_size	Specifies the maximum number of application objects' metadata to cache. Possible values are any integer greater than or equal to 1. The default value is 200.	agent_max_ao_cache_size=200

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
`agent_max_co_cache_size`	Specifies the maximum number of common objects' metadata to cache. Possible values are any integer greater than or equal to `1`. The default value is `100`.	`agent_max_co_cache_size=100`
`agent_max_message_metadata_cache_size`	Specifies the maximum number of messages' metadata (publish/subscribe and invoke/implement) to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_message_metadata_cache_size=200`
`agent_max_dvm_table_cache_size`	Specifies the maximum number of DVM tables to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_dvm_table_cache_size=200`
`agent_max_lookup_table_cache_size`	Specifies the maximum number of lookup tables to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_lookup_table_cache_size=200`
`agent_max_queue_size`	Specifies the maximum size to which internal OracleAS InterConnect message queues can grow. Possible values are any integer greater than or equal to `1`. The default value is `1000`.	`agent_max_queue_size=1000`
`agent_persistence_queue_size`	Specifies the maximum size to which internal OracleAS InterConnect persistence queues can grow. Possible values are any integer greater than or equal to `1`. The default value is `1000`.	`agent_persistence_queue_size=1000`
`agent_persistence_cleanup_interval`	Specifies how often to run the persistence cleaner thread (in milliseconds). Possible values are any integer greater than or equal to `30000` milliseconds. The default value is `60000`.	`agent_persistence_cleanup_interval=60000`
`agent_persistence_retry_interval`	Specifies how often the persistence thread retries when it fails to send an OracleAS InterConnect message. Possible values are any integer greater than or equal to `5000` milliseconds. The default value is `60000`.	`agent_persistence_retry_interval=60000`
`agent_pipeline_to_hub`	Specifies how to turn on or off the pipeline for messages from the Bridge towards the hub. If you set the pipeline to `false`, the file persistence is not used in that direction.	`agent_pipeline_to_hub=false`
`agent_pipeline_from_hub`	Specifies how to turn on or off the pipeline for messages from the hub towards the Bridge. If you set the pipeline to `false`, the file persistence is not used in that direction.	`agent_pipeline_from_hub=false`

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
`service_path`	Windows only. Specifies the value to which to set the environment variable `PATH`. The `PATH` variable is set to the specified value before forking the Java VM. Typically, list all directories here that contain all necessary DLLs. Possible values are the valid `PATH` environment variable setting. There is no default value.	`service_path=%JREHOME%\bin;D:\oracle\ora904\bin`
`service_classpath`	Windows only. Specifies the class path used by the adapter Java VM. If a custom adapter is developed and, as a result, the adapter is to pick up any additional jars, add the jars to the existing set of jars being picked up. Possible values are the valid class path. There is no default value.	`service_classpath=D:\oracle\ora904\oai\904\lib\oai.jar;%JREHOME%\lib\rt.jar;D:\oracle\ora904\jdbc\classes12.zip`
`service_class`	Specifies the entry class for the Windows service. A possible value is `oracle/oai/agent/service/AgentService`. There is no default value.	`service_class=oracle/oai/agent/service/AgentService`
`service_max_java_stack_size`	Windows only. Specifies the maximum size to which the Java VM's stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	`service_max_java_stack_size=409600`
`service_max_native_stack_size`	Windows only. Specifies the maximum size to which the Java VM's native stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	`service_max_native_stack_size=131072`
`service_min_heap_size`	Windows only. Specifies the minimum heap size for the adapter Java VM. Possible values are the valid Java VM heap sizes. The default value is `536870912`.	`service_min_heap_size=536870912`
`service_max_heap_size`	Windows only. Specifies the maximum heap size for the adapter Java VM. Possible values are any valid Java VM heap sizes. The default value is `536870912`.	`service_max_heap_size=536870912`
`service_num_vm_args`	Windows only. Specifies the number of `service_vm_arg`*`number`* parameters specified. Possible values are the number of `service_vm_arg`*`number`* parameters. There is no default value.	`service_num_vm_args=1`

Table 2–7 Agent Connection Parameters

Parameter	Description	Example

`service_vm_arg`*`number`*	Windows only. Specifies any additional arguments to the Java VM. For example, to retrieve line numbers in any of the stack traces, set `service_vm_arg1=java.compiler=NONE`. If a list of arguments to specify exists, use multiple parameters as shown in the example by incrementing the last digit starting with `1`. Be sure to set `service_num_vm_args` correctly. Possible values are any valid Java VM arguments. There is no default value.	`service_vm_arg1=java.compiler=NONE`
		 `service_vm_arg2=oai.adapter=database`

`corba_port_number`	The CORBA port number on which the adapter CORBA service listens. Generally, this port is allocated dynamically. However, it can be configured to enable access across firewall.	`corba_port_number=14000`
`encoding`	Character encoding for published messages. The adapter uses this parameter to generate encoding information in encoding tag of transformed OracleAS InterConnect message. OracleAS InterConnect represents messages internally as an XML document. The default encoding of the XML document is `UTF-8`. However, this encoding can be configured using this parameter, which is typically used when the OracleAS InterConnect message consists of characters not supported by `UTF-8` and when the `XMLParser` is unable to handle them.	`encoding=JA16SJIS`

Table 2–7 Agent Connection Parameters

Parameter	Description	Example

`nls_date_format`	Format for date fields expressed as string. The default date format is `EEE MMM dd HH:mm:ss zzz yyyy`. For the meaning of this string, see the list of reserved characters in Table 2–8.	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_country`	This parameter is a valid ISO Country Code. These upper-case and two-letter codes are defined by ISO-3166. You can find a full list of these codes at a Web site, such as, `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html` The default Country code is `US`. **Note**: This parameter specifies date format. It is applicable for the date format only.	`US`
`nls_language`	This parameter is a valid ISO Language Code. These lower-case and two-letter codes are defined by ISO-639. You can find a full list of these codes at a Web site, such as, `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt` The default language code is `en`. **Note**: This parameter specifies date format. It is applicable for the date format only.	`nls_language=en`

Table 2–8 shows the reserved characters used to specify the value of the `nls_date_format` parameter. Using these characters, you can construct a pattern to define date formats.

Table 2–8 Reserved Characters for the Value of the nls_date_format Parameter

Letter	Description	Example
`G`	Era designator	`AD`
`y`	Year	`1996; 96`
`M`	Month in year	`July; Jul; 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; 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`
`s`	Second in minute	`55`
`S`	Millisecond	`978`
`G`	Era designator	`AD`

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. A value must be specified and cannot be modified later. A possible value is `oracle.oai.agent.adapter.technology.TechBridge`. There is no default value.	`bridge_class=oracle.oai.agent.adapter.technology.TechBridge`
`ota.send.endpoint`	Defines the sending endpoint URL for the HTTP adapter. There is no default value. The URL is of the form: `http(s)://`*`hostname`*`:`*`port/path`*	`ota.send.endpoint=http://site.com:8888/servlet/inbound`
`ota.receive.endpoint`	Defines the receiving endpoint URL for the HTTP adapter. There is no default value. The URL is of the form: `http(s)://`*`hostname`*`:`*`port/path`*	`ota.receive.endpoint=http://site.com:8888/servlet/inbound`
`ota.type`	Defines the message payload type the HTTP adapter handles for both incoming and outgoing messages. The options are `XML`, `XML_NVP`, or `D3L`. The default value is `XML`.	`ota.type=XML`
`ota.d3ls`	Specifies the list of D3L XML files used by this 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`
`http.sender.timeout`	Times out an HTTP connection. The unit is in milliseconds. The default is set to `60000` milliseconds (60 seconds).	`http.sender.timeout= 10000` (Sets timeout to 10 seconds)
`http.sender.authtype`	Set if authentication is needed. The valid options are `basic` or `digest`. There is no default value.	`http.sender.authtype= basic`
`http.sender.realm`	Specifies the realm for the authentication scheme. There is no default value.	`http.sender.realm=ipt`
`http.sender.username`	Specifies the authentication username. There is no default value.	`http.sender.username= joe`

Table 2–9 HTTP Adapter-Specific Parameters

Parameter	Description	Example
`http.sender.password`	Specifies the password used in the `sender.password` authentication. There is no default value. This password can also be encrypted by running the encrypt tool and renaming this parameter to `encrypted_http.sender.password`. **See Also:** "How do I make the adapter.ini file password parameters secure?" on page 4-6 for instructions on encrypting the user password	`http.sender.password=httpuser`
`http.sender.proxy_host`	Specifies the proxy hostname. There is no default value.	`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. There is no default value.	`http.sender.proxy_port=80`
`http.sender.wallet_location`	Needed if SSL is used. This specifies the path and name of the exported wallet file (not `.p12` file). There is no default value.	`http.sender.wallet_location=/private/foo/certdb.txt`
`http.sender.wallet_password`	Needed if SSL is used. This specifies the password for the Oracle Wallet Manager. There is no default value. This password can also be encrypted by running the encrypt tool and renaming this parameter to `encrypted_http.sender.wallet_password`. **See Also:** "How do I make the adapter.ini file password parameters secure?" on page 4-6 for instructions on encrypting the wallet password	`http.sender.wallet_password=walletuser`

Table 2–9 HTTP Adapter-Specific Parameters

Parameter	Description	Example

`http.sender.cipher_suites`	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`
		`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 by the HTTP sender. The default value is: `oracle.oai.agent.adapter.technology.HTTPDefaultSenderCustomizer`	`http.sender.customizer_class=MyHTTPSenderCustomizer`
`http.receiver.registry_port`	Specifies the RMI port used by the HTTP receiver. The default is `9901`.	`http.receiver.registry_port=9901`
`http.receiver.instance_name`	Specifies the instance name of the HTTP receiver. The default is `oai`. If the default value is not used, the `instanceName` of the initial parameter of the transport servlet must be modified to match this instance name.	`http.receiver.instance_name=oai`
`http.receiver.customized_class`	Specifies the class name for customizing the HTTP response.	`http.receiver. customized_class=MyBanner`
`http.receiver.customizer_class`	Specifies the class name for customizing by the HTTP sender. The default value is: `oracle.oai.agent.adapter.technology.HTTPDefaultSenderCustomizer`	`http.receiver.customizer_class=MyHTTPReceiverCustomizer`

3 Design Time and Runtime Concepts

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

This chapter contains these topics:

HTTP Adapter Design Time Concepts

The HTTP adapter can handle XML and data definition description language (D3L) structured payload. For example:

- Pure XML data—string beginning with `<?xml . . .`
- Fixed layout, typically binary data described by a D3L XML file

> **See Also:** *Oracle Application Server InterConnect User's Guide*, Appendix B, for additional information on D3L

XML Payload Type

You can import a document type definition (DTD) in iStudio that determines how the HTTP adapter parses a received XML document into an OracleAS InterConnect application view event. In addition, the DTD describes how an inbound application view message is converted into an XML document. Use the message type option XML when defining a new integration point in any of the Event Wizards.

You must also ensure that the parameter `ota.type` 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 XML payload mode, no translations are performed on the messages (between native view and application view) sent or received through the HTTP adapter. This is apart from the implied straight ASCII to Java object conversion (parsing). Any Extensible Stylesheet Language transformations (XSLT) should be performed before sending or receiving an XML document to or from OracleAS InterConnect.

D3L Payload Type

The HTTP adapter supports both XML and D3L datatypes. The HTTP adapter converts and translates application view messages to native format and vice versa.

An application based on the HTTP adapter can use the iStudio Message Type D3L and the iStudio D3L Data Type Import option when importing a datatype. In doing so, 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 datatypes.

See Also: *Oracle Application Server InterConnect User's Guide*, Appendix B, for additional information on D3L and common view data types

HTTP Adapter Runtime Concepts

This section describes the key runtime components of the HTTP adapter.

This section contains these topics:

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

See Also: *Oracle Application Server InterConnect User's Guide*, Appendix B, for an example involving an Oracle adapter, the Advanced Queuing adapter, and D3L

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 InterConnect.

In a typical deployment, the servlet runs in Oracle Application Server Containers for J2EE (OC4J). The servlet processes the HTTP client requests and relays them to the HTTP receiver through remote method invocation (RMI). Upon receiving the message, the HTTP receiver passes the message to the HTTP bridge.

The HTTP bridge uses the D3L XML file based on name/value pair 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 native format message into an OracleAS InterConnect message object and translate to an application view event. The agent converts the application view event into a common view event and sends it to OracleAS InterConnect for further routing and processing.

Once the message is successfully sent to OracleAS InterConnect, the HTTP adapter returns a `200` message acknowledgment.

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

See Also:

- *Oracle Application Server 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 1-3
- "HTTP Adapter-Specific Parameters" on page 2-22

HTTP Sender

The HTTP adapter supports sending outgoing messages from OracleAS InterConnect to multiple HTTP endpoints. The multiple endpoints feature provides flexibility for sending messages to different 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 through the Modify Fields button of the Subscribe Wizard - Define Application View dialog. Once the association of endpoint and event is established, the message from the subscribing event is sent out to the HTTP endpoint.

For example, the metadata in Table 3–1 is associated with an event called `sendOrder` that sends an order to an HTTP server at `foo.com` with a path of `/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 HTTP adapter's sending endpoint

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

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 perform the translation of common view object to native format message. The native format message is then sent through the HTTP transport layer to an HTTP endpoint. The properties for the HTTP sender are defined in the `adapter.ini` file and take the form of `http.sender.*`.

Note: When using the multiple-endpoint feature with `XML` data type, you must choose the event type of `Generic`, instead of `XML`. Using the `Generic` event type allows you to enter the metadata for the endpoints via the Modify Fields feature associated with iStudio.

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

See Also:

- Figure 1–2, "Outgoing Messages" on page 1-4
- "HTTP Adapter-Specific Parameters" on page 2-22
- Chapter 4 of the *Oracle Application Server InterConnect User's Guide* for information on adding transport properties as metadata in iStudio

HTTP Adapter Message Format

This section describes how to extract or send messages 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.

D3L Payload Type

You must ensure that the `ota.type` parameter in the `adapter.ini` file is set to `D3L` to use this payload type. The HTTP adapter expects to receive 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 with the `POST` method to either of the following:

- 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

XML Payload Type

You must ensure that the `ota.type` parameter in the `adapter.ini` file is set to `XML` to use this payload type. The sending and receiving operation for the XML payload type is similar to D3L. With XML, the D3L transformation is not performed.

XML_NVP (XML Name-Value Pair) You must ensure that the `ota.type` parameter in the `adapter.ini` file is 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 InterConnect object. During the sending operation, the adapter packages the name-value pair and sends it through the `POST` method.

> **See Also:** The `ota.type` parameter description on page 2-22 for information on setting the payload message type in the `adapter.ini` file

HTTP Message Headers

Example 3–1 shows the HTTP message header types and data sent by the HTTP adapter:

Example 3–1 HTTP 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 lists and describes key `OAI-*` headers.

Table 3–2 `OAI-*` ***Headers***

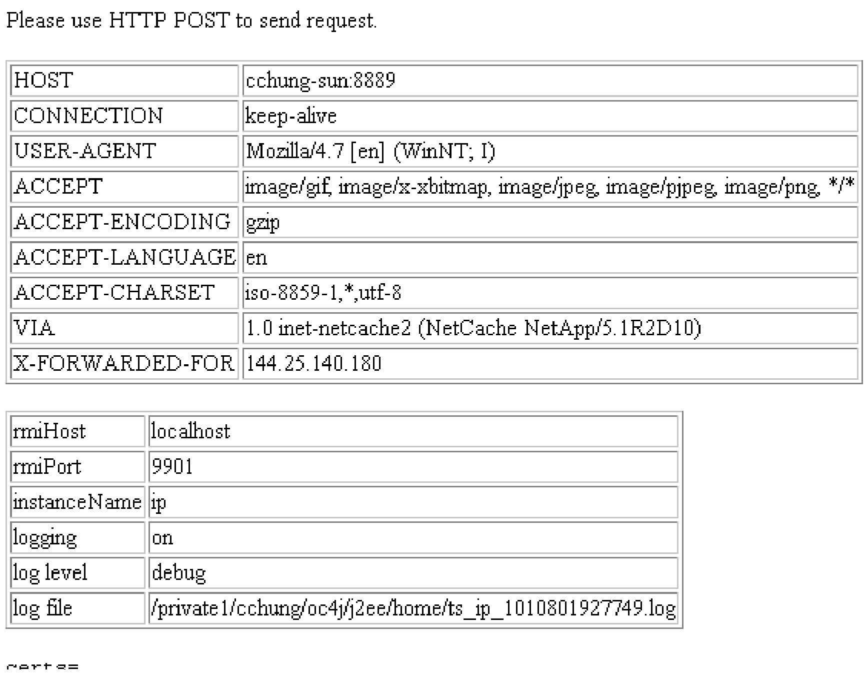
Header	Description
`OAI-MV`	Message version to which this message corresponds, as created in iStudio
`OAI-T`	Possible values are: `0` (publish) `1` (request) `2` (reply) Only publish is supported in this release.
`OAI-BO`	Business object name to which this message corresponds
`OAI-EV`	Event version to which this message corresponds, as created in iStudio
`OAI-EN`	OracleAS InterConnect event name
`OAI-APPLICATION`	HTTP adapter application name

HTTP Receiver Diagnostics

This section describes how to determine if the HTTP receiver is functioning properly.

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, 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 is useful for identifying information that the servlet reads from the `web.xml` file. The rmiHost, rmiPort and instance name have to match the corresponding parameters in the `adapter.ini` file.

Customizing the HTTP Adapter

You can customize some of the adapter behaviors by implementing the following two interfaces:

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

You can use the `ReceiverCustomizer` interface to customize the message that the HTTP adapter receives. The `ReceiverCustomizer` interface replaces the following customizing interface, which is from the previous release and is currently deprecated, `oracle.oai.agent.adapter.transport.basic.HTTPCustomizedResponse`.

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

ReceiverCustomizer Interface

File Structure

The following is the file structure of this 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);
}
```

File Summary

The following table summarizes the `ReceiveCustomizer` interface

Method	Description
`customizeTransport Message();`	This method allows the user to customize the transport message received by the adapter, Message. It contains the following parameters: agent—Used to log a message. receiverType—Provides information on the type of adapter. transportMessage—Used to customize the transport message received by the adapter.

Method	Description
`createReplyMessage ();`	This method creates a reply message, Message, based on the status and the message received. It contains the following parameters: agent—Used to log a message. status—The status of the message process. If the value is TransportResponse.TRANSPORT_ACK, the message is processed successfully. If the value is TransportResponse.TRANSPORT_ERROR, the message is processed unsuccessfully. receivedTransportMessage—The transport message is received by the adapter. This parameter is used to transport headers in the transport message to create a meaningful HTTP message. The return string contains the reply message. This method is included for backward compatibility.

Example 3–2 Example of ReceiverCustomizer

This `MyReceiverCustomizer` class removes the first line in the native message.

```
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 InterConnect does not understand.

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

Note: 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(int status) {
      String response = Message has unknown status.";
      switch (status) {
```

> **Note:** OracleAS InterConnect indicates to the transport layer that the message has been processed successfully.

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

> **Note:** OracleAS 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.";
}
```

Example 3–3 List of Methods for the TransportMessage Class

This example provides a list of methods users may choose for the `TransportMessage` class.

Method	**Description**
`public String toString();`	Dump message and headers.
`public void setTransportHeader(String name, String value);`	Set a transport specific header.
`public Properties getTransportHeaders();`	Get all transport specific header and return a Properties object that contains all the transport headers.
`public void setBody(String body) throws TransportException;`	Set the body of the message. The body type will be set to STRING. Parameter includes: `body`—body of the message. It throws a `TransportException`.

Method	Description
`public void setBody(InputStream in) throws TransportException;`	Set the body of the message. The body type will be set to BYTES. Parameter includes: `InputStream`—Contains the message. It throws a `TransportException`.
`public String getBodyAsString();`	Get the body of the message as String object. Return the message in String object.
`public byte[] getBodyAsBytes();`	Get the body of the message as byte array. Return the message in byte[].
`public InputStream getBodyAsInputStream();`	Get the body of the message and return an `InputStream` object representing the body of the message.

HTTPSenderCustomizer Interface

You can use the `THHPSenderCustomizer` 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.

SenderCustomizer Interface

File Structure

The following is 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);

}
```

File Summary

The following table summarizes the `customizerTransportMessage` method.

Method	Description
`customizerTransportMessage();`	This method specifies how to customize the transport message for transport sender. The adapter creates a `TransportMessage` for the transport layer to send based on the `MessageObject` sent by OracleAS InterConnect. You can use this method to further customize the transport message to be sent out by the transport layer. This method contains the following parameters: `agent`—Used to log messages. `transportMessage`—Indicates the `TransportMessage` object that the adapter has created for sending. `mobj`—Indicates the `MessageObject` from OracleAS InterConnect. `aobj`—Indicates the `AttributeObject` from OracleAS InterConnect. This method does not return anything. You can change the payload with the `transportMessage` parameter.

HTTPSenderCustomizer Interface

The following is the 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{

}
```

Starting the HTTP Adapter

On UNIX, start the HTTP adapter using the `start` script located in the following directory:

ORACLE_HOME/oai/9.0.4/adapters/*Application*

Type **start**, then press **Enter**.

On Windows, start the HTTP adapter from the Services window available from the Start menu.

1. Access the Services window from the Start menu:

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

   The Services window appears.

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

On...	Choose...
Windows NT	Choose Start.
Windows 2000	Right-click the service and choose Start from the menu that appears.

See Also: "HTTP Adapter Configuration Parameters" on page 2-11 for the location of the `start` script

Log File Example of Successfully Started HTTP Adapter

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

`timestamp_in_milliseconds`

The following file displays information about an HTTP adapter that started successfully:

```
D:\oracle\ora904\oai\9.0.4\adapters\httpapp>D:\oracle\ora904\oai\9.0.4\bin\JavaS
ervice.exe -debug "Oracle OAI Adapter 9.0.4
-httpapp" D:\oracle\ora9041\oai\9.0.4\adapters\httpapp adapter.ini
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.
```

Stopping the HTTP Adapter

On UNIX, stop the HTTP adapter using the `stop` script located in the following directory named after the Oracle HTTP application.

```
ORACLE_HOME/oai/9.0.4/adapters/Application
```

Type **stop**, then press **Enter**.

On Windows, stop the HTTP adapter from the Services window available from the Start menu.

1. Access the Services window from the Start menu:

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

   The Services window appears.

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

On...	Choose...
Windows NT	Choose Stop.
Windows 2000	Right-click the service and choose Stop from the menu that appears.

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

See Also: "HTTP Adapter Configuration Parameters" on page 2-11 for the location of the `stop` script

HTTP Adapter Error Codes

The HTTP adapter returns the standard HTTP status codes as outlined in Request For Comments (RFC) 2068. See Section 6.1.1 "Status Code and Reason Phrase" of this document.

See Also: The following URL for access to RFC 2068:

`http://www.w3.org/Protocols/`

4 Frequently Asked Questions

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

This chapter contains this topic:

- Troubleshooting Questions

Troubleshooting Questions

The following questions address troubleshooting issues for the HTTP adapter.

How do I know the HTTP adapter started properly?

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

On...	Go to...
UNIX	*ORACLE_HOME*/oai/9.0.4/adapters/*Application*/logs/*timestamp_in_milliseconds*
Windows	*ORACLE_HOME*\oai\9.0.4\adapters\Application\logs\timestamp_in_milliseconds

where *Application* is the value you defined in Step 3 on page 2-3 and *timestamp_in_milliseconds* is the directory. If no exceptions are listed, the adapter 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 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 the information from iStudio (the information that is stored in the repository) locally for better performance in a production environment. If you change something in iStudio and want to see the change in the runtime environment, you need to perform the following procedures:

To see iStudio changes in the runtime environment:

1. Stop the affected adapters.
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, edit the parameters in the following file:

On...	Go to...
UNIX	`ORACLE_HOME/oai/9.0.4/adapters/Application/adapter.ini`
Windows	`ORACLE_HOME\oai\9.0.4\adapters\Application\adapter.ini`

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

See Also: "Hub.ini Parameter File" on page 2-12 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, 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 (see the next question for details).

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

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

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

   On UNIX:

```
% cd ORACLE_HOME/oai/9.0.2/bin
% copyAdapter <oldAdapterName> <newAdapterName>
```

On Windows:

```
c:\> cd ORACLE_HOME\oai\9.0.2 bin
c:\< copyAdapter <oldAdapterName> <new AdapterName>
```

2. Change the parameters in the `adapter.ini` file for the new adapter. In particular, make sure the following parameters in the new `adapter.ini` file are different from the `adapter.ini` file for the existing HTTP adapter:

   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 machine.

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/oai/9.0.2/adapters/<newAdapterName>/webapps/WEB-INF
   ```

   On Windows:

   ```
   ORACLE_HOME\oai\9.0.2\adapters\<newAdapterName>\webapps\WEB-INF
   ```

   ■ Change the rmi port to match the value entered in Step 2d.

     * Change the following entry in the `web.xml` file:

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

       to:

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

4. Change the following entry in the `application.xml` file in the `ORACLE_HOME\oai\9.0.2\adapters\<your new http app name>\webapps\META-INF` directory:

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

   to:

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

5. Prepare the java archive parameter (`oai1.ear`):

   On UNIX:

   ```
   % cd ORACLE_HOME/oai/9.0.2/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\oai\9.0.2\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 has been created called `oai1.ear` which is ready for deployment.

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

7. Restart the HTTP server. Verify if the new receiving endpoint is functioning by entering the URL used in Step 2b in your browser. If the servlet is deployed correctly, a diagnostic page appears.

How do I make the adapter.ini file password parameters secure?

In order to encrypt password values specified in the `adapter.ini` file, perform the following steps:

To encrypt password values:

1. Locate the password value to encrypt.
2. Run the encrypt utility to encrypt the password value. The encrypt utility is located in the *ORACLE_HOME*`/oai/9.0.4/bin` directory for UNIX and the *ORACLE_HOME*`\oai\9.0.4\bin` directory for Windows. For example, to encrypt the `http.sender.password` parameter, enter the following:

   ```
   encrypt password
   ```

3. Prefix the name of the parameter in the `adapter.ini` file with `encrypted_`:

   ```
   encrypted_http.sender.password
   ```

4. Replace the value with the new encrypted value created in Step 2. For example, if you want to encrypt the password for the parameter `http.sender.password`, replace the line:

   ```
   http.sender.password=HTTPuser
   ```

   with the value you received from running the encrypt tool in Step 2:

   ```
   encrypted_http.sender.password=112411071071106510801094108410731070107110811069
   ```

   Note: You can also encrypt the `http.sender.wallet_password` parameter by performing these instructions.

A

adapter.ini Example File

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

This appendix contains this topic:

- adapter.ini Example File

See Also: "HTTP Adapter Configuration Parameters" on page 2-11 for additional information on `adapter.ini` configuration parameters

adapter.ini Example File

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

```
#include <../../hub/hub.ini>
// *************
// ** Adapter **
// *************
// Application (as created in iStudio) that this Adapter corresponds to.
application=HTTPapp1
// Partition (as created in iStudio) that this Adapter corresponds to.
partition=
// If you want to have multiple Adapter instances for the 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=112411071071106510801094108410731070107110811069

// 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=112411071071106510801094108410731070107110811070
//
//   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
since 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