# Oracle® Application Server Integration InterConnect

Adapter for HTTP Installation and User's Guide

10*g* Release 2 (10.1.2)

**B14074-02**

December 2005

ORACLE®

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

B14074-02



Primary Author: Rima Dave

Contributing Author: Vimmy K Raj

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

# Contents

## 3 Design-Time and Run-Time Concepts



## A Frequently Asked Questions



## B Example of the adapter.ini File

## Index

# 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. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

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

### Accessibility of Code Examples in Documentation

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

### Accessibility of Links to External Web Sites in Documentation

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

### TTY Access to Oracle Support Services

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

## Related Documents

For more information, refer to these Oracle resources:

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

## Conventions

The following text conventions are used in this document:

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

# 1 Introduction

This 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***

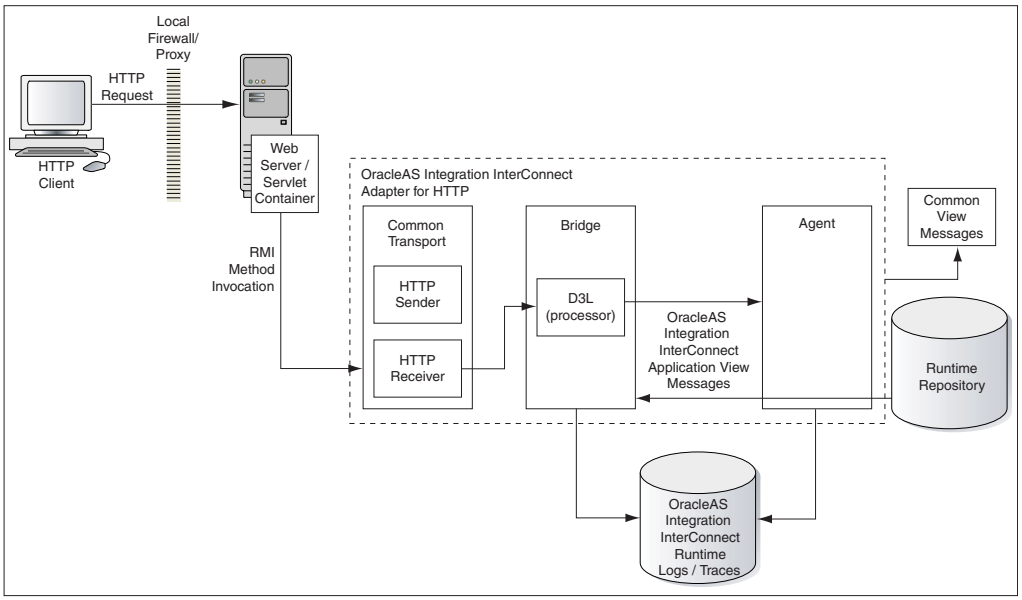
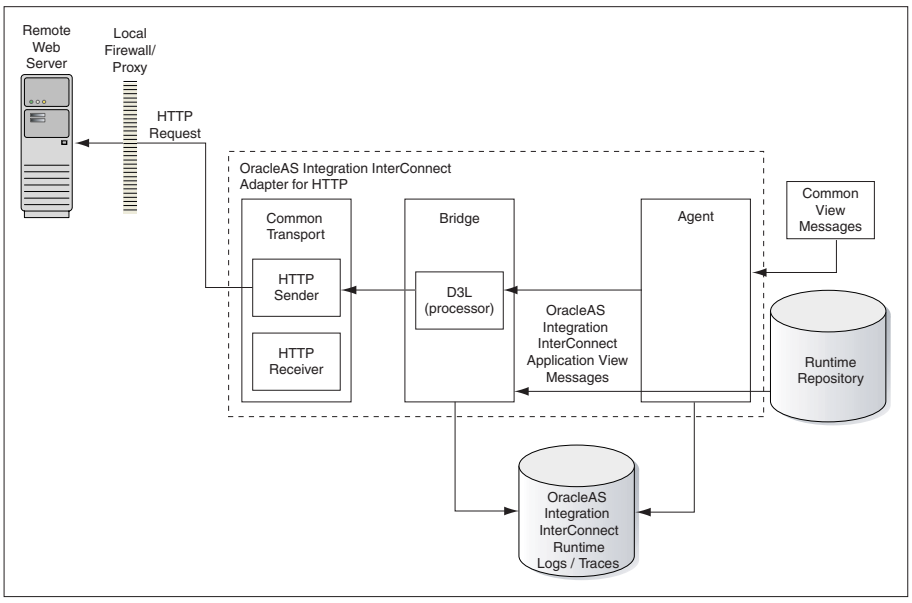



Figure 1–2 depicts the data flow of outgoing messages from OracleAS Integration InterConnect to a remote Web server. Outgoing messages are sent from an HTTP sender object in the adapter to the remote Web server.

***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 installing the HTTP adapter.

*Table 1–1 Hardware Requirements*

| Hardware | Windows | UNIX |
|---|---|---|
| Disk Space | 400 MB | 400 MB |
| Memory | 512 MB | 512 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 installing the HTTP adapter.

*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<br>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.
- Enables 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:



## 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:



### 2.1.1 Preinstallation Tasks

Refer to 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 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 wizard, select **OracleAS Integration InterConnect Adapter for HTTP 10.1.2.0.2** and click **Next**.
2. The Set Oracle Wallet Password page is displayed. Enter and confirm the password on the page, which will be used to manage 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`.

   **Note:** You can change the application name in iStudio after installation. In such case, you need to specify the password corresponding to new application name in the Oracle Wallet.

   For more informtion, refer to the following sections in Appendix A, "Frequently Asked Questions":

   - Section A.3, "The HTTP adapter is not starting. What could be the reason?"
   - Section A.7, "How do I secure my passwords?"

6. Click **Next**. The Specify HTTP Adapter Usage page 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 page that is displayed next 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 following table lists the platform and the directory in which the HTTP adapter will be installed.

| 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

The 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 2-12 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 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"
- Section A.7, "How do I secure my passwords?" of Appendix A, "Frequently Asked Questions" 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 Section A.7, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions" for more details on how to modify and retrieve the password by 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"
- Section A.7, "How do I secure my passwords?" of Appendix A, "Frequently Asked Questions" for instructions on how to modify and retrieve the password

#### 2.1.3.6 Customizing the Receiving Endpoints

To customize the 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 Installing Multiple HTTP Adapters in the Same Oracle Home

To install multiple instances of the HTTP adapter in same Oracle home, perform the following:

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
   ```

   **Note:** The `copyAdapter` script is copied to the following `bin` directory only during Hub installation:

   - UNIX: `ORACLE_HOME/integration/interconnect/bin`
   - Windows: `ORACLE_HOME\integration\interconnect\bin`

   If you need to use this script to create multiple adapters on a spoke computer, then copy the script to the `bin` directory on the spoke computer, and edit the script to reflect the new Oracle home.

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.

9. 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.

After running the `copyAdapter` script, If you want to manage or monitor the newly installed adapter through Oracle Enterprise Manager 10*g* Application Server Control Console, then you need to modify the `opmn.xml` file by adding information about the new instance. For example, you have created a new instance of the HTTP adapter

myHTTPApp1 by using the copyAdapter script. To manage the myHTTPApp1 adapter through Enterprise Manager, perform the following:

1. Navigate to the *MiddleTier*\bin directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

2. Next, specify the information about this new instance in the opmn.xml file located in the *ORACLEMIDDLETIER_HOME*/opmn/conf directory as follows:

   ```
   <process-type id="myHTTPApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/myHTTPApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myHTTPApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
                       <data id="class-name"
                        value="oracle.oai.agent.proxy.ShutdownAgent"/>
                       <data id="application-parameters"
                        value="persistence/Agent.ior"/>
                   </category>
               </module-data>
          </process-set>
   </process-type>
   ```

   The opmn.xml file would appear like this:

   ```
   <process-type id="myHTTPApp" module-id="adapter" working-dir="$ORACLE
   _HOME/integration/interconnect/adapters/myHTTPApp" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myHTTPApp" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
                       <data id="class-name"
                value="oracle.oai.agent.proxy.ShutdownAgent"/>
                <data id="application-parameters"
                 value="persistence/Agent.ior"/>
            </category>
        </module-data>
     </process-set>
   </process-type>

   <process-type id="myHTTPApp1" module-id="adapter" working-dir="$ORACLE
   _HOME/integration/interconnect/adapters/myHTTPApp1" status="enabled">
   ```

```
        <start timeout="600" retry="2"/>
        <stop timeout="120"/>
        <port id="icadapter_dmsport_range" range="15701-15800"/>
        <process-set id="myHTTPApp1" restart-on-death="true" numprocs="1">
         <module-data>
           <category id="start-parameters">
               <data id="java-parameters" value="-Xms8M"/>
               <data id="class-name"
                value="oracle.oai.agent.service.AgentService"/>
           </category>
           <category id="stop-parameters">
                <data id="java-parameters" value="-mx64m"/>
                <data id="class-name"
                 value="oracle.oai.agent.proxy.ShutdownAgent"/>
                <data id="application-parameters"
                 value="persistence/Agent.ior"/>
           </category>
         </module-data>
        </process-set>
</process-type>
```

3. Save the `opmn.xml` file.
4. Navigate to the *`MiddleTier`*`\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

5. You can start the `myHTTPApp1` adapter by using the following command

   ```
   opmnctl startproc ias-component="InterConnect" process-type="myHTTPApp1"
   ```

6. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

7. Login to the Oracle Enterprise Manager 10*g* Application Server Control Console to view and manage the newly installed or copied adapter. For information about how to use Oracle Enterprise Manager 10*g* Application Server Control Console , refer to the *Oracle Application Server Integration InterConnect User's Guide*

---

**Note:** While installing multiple adapters in the same computer, the copyadapter script does not create entries for the new adapter's password in the Oracle Wallet. You need to manually create a password for this new adapter using the Oracle Wallet Manager. To store the password in Oracle Wallet, use the following format:

```
ApplicationName/password
```

The number of entries is dependent on the type of adapter. For example, Database adapter needs two entries whereas AQ Adapter needs only one entry. For more information about how to manage your passwords in Oracle Wallet, refer to Section A.7, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions"

---

## 2.3 Configuring the HTTP Adapter

After installing the HTTP adapter, you can configure it according to your requirements. 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 log.xml log file. |
| persistence | The messages are made available in this directory. Do not edit this directory or its files. |

### 2.3.1 HTTP Adapter Ini File Settings

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

- hub.ini Files
- adapter.ini Files

#### 2.3.1.1 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 gives a description and an example for each parameter.

***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 which can range from 2 to total number of nodes in cluster. 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_instance`*x* | 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` |

#### 2.3.1.2 adapter.ini Files

The agent component of the HTTP adapter reads the `adapter.ini` file at run time to access information on configuring the HTTP adapter parameter. Table 2–7 gives a description and an example for each parameter.

***Table 2–7 adapter.ini Parameters***

| Parameter | Description | Example |
|---|---|---|
| `agent_admin_port` | Specifies the port through which the adapter can be accessed through firewalls.<br>Possible value: A valid port number<br>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.<br>Possible values: `true` or `false`<br>Default value: `false`<br>**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.<br>Possible values:<br>■ `startup`: Cache all DVM tables at startup. This may take a while if there are a lot of tables in the repository<br>■ `demand`: Cache tables as they are used<br>■ `none`: No caching. This slows down performance<br>Default value: `demand` | `agent_dvm_table_caching=demand` |
| `agent_log_level` | Specifies the amount of logging necessary.<br>Possible values:<br>■ `0`= errors only<br>■ `1`= status and errors<br>■ `2`= trace, status, and errors<br>Default value: `1` | `agent_log_level=2` |
| `agent_lookup_table_caching` | Specifies the lookup table caching algorithm.<br>Possible values:<br>■ `startup`: Cache all lookup tables at startup. This may take a while if there are a lot of tables in the repository<br>■ `demand`: Cache tables as they are used<br>■ `none`: No caching. This slows down performance.<br>Default value: `demand` | `agent_lookup_table_caching=demand` |
| `agent_max_ao_cache_size` | Specifies the maximum number of application object metadata to cache.<br>Possible value: An integer greater than or equal to `1`<br>Default value: `200` | `agent_max_ao_cache_size=200` |
| `agent_max_co_cache_size` | Specifies the maximum number of common object metadata to cache.<br>Possible value: An integer greater than or equal to `1`<br>Default value: `100` | `agent_max_co_cache_size=100` |

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

| Parameter | Description | Example |
|---|---|---|
| `agent_max_dvm_table_cache_size` | Specifies the maximum number of DVM tables to cache.<br>Possible value: An integer greater than or equal to `1`<br>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.<br>Possible value: An integer greater than or equal to `1`<br>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.<br>Possible value: An integer greater than or equal to `1`<br>Default value: `200` | `agent_max_message_metadata_cache_size=200` |
| `agent_max_queue_size` | Specifies the maximum size to which internal OracleAS Integration InterConnect message queues can grow.<br>Possible value: An integer greater than or equal to `1`<br>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.<br>Possible value: A valid Oracle Advanced Queue message selector string (such as `'%,aqapp,%'`)<br>Default value: None | `agent_message_selector=%,aqapp,%` |
| `agent_metadata_caching` | Specifies the metadata caching algorithm.<br>Possible values:<br>■ `startup`: Cache everything at startup. This may take time if there are a lot of tables in the repositor<br>■ `demand`: Cache metadata as it is used<br>■ `none`: No caching. This slows down performance<br>Default value: `demand` | `agent_metadata_caching=demand` |
| `agent_persistence_cleanup_interval` | Specifies how often to run the persistence cleaner thread in milliseconds.<br>Possible value: An integer greater than or equal to `30000` milliseconds<br>Default value: `60000` | `agent_persistence_cleanup_interval=60000` |
| `agent_persistence_queue_size` | Specifies the maximum size of internal OracleAS Integration InterConnect persistence queues.<br>Possible value: An integer greater than or equal to `1`<br>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.<br>Possible value: An integer greater than or equal to `5000` milliseconds<br>Default value: `60000` | `agent_persistence_retry_interval=60000` |

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

| Parameter | Description | Example |
|---|---|---|
| agent_pipeline_from_hub | Specifies whether to activate 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.<br>Possible value: true, false<br>Default value: false | agent_pipeline_from_hub=false |
| agent_pipeline_to_hub | Specifies whether to activate 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.<br>Possible value: true, false<br>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.<br>Possible value: A string built using the application name (parameter:application) concatenated with the instance number (parameter:instance_number).<br>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.<br>Possible value: The application name (parameter:application) concatenated with the instance number (parameter:instance_number)<br>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.<br>Possible value: A valid Oracle Advanced Queue subscriber name<br>Default value: None | agent_subscriber_name=httpapp |
| agent_throughput_measurement_enabled | Specifies if the throughput measurement is enabled. Set this parameter to true to activate all throughput measurements.<br>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.<br>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.<br>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.<br>Default value: None | agent_use_custom_hub_dtd=false |

*Table 2–7 (Cont.) adapter.ini 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 while creating metadata.<br>Possible value: An alphanumeric string<br>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.<br>Possible value: A valid character encoding<br>Default value: UTF-8<br>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.<br>Possible value: A URL<br>Default value: The URL of the current user directory | external_dtd_base_url=file://C:\InterConnect10_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.<br>Possible value: An integer greater than or equal to 1<br>Default value: None | instance_number=1 |
| nls_country | Specifies the ISO country code. The codes are defined by ISO-3166.<br>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<br>Default value: US<br>**Note**: This parameter specifies date format. It is applicable only for the date format. | nls_country=US |
| nls_date_format | Specifies the format for a date field expressed as a string.<br>Possible value: Any valid date format pattern as shown in Table 2–8 for the definitions of the format characters.<br>Default value: EEE MMM dd HHmmss zzz yyyy | Date format pattern dd/MMM/yyyy can represent 01/01/2003.<br>nls_date_format=dd-MMM-yy<br>Multiple date formats can be specified as num_nls_formats=2<br>nls_date_format1=dd-MMM-yy<br>nls_date_format2=dd/MMM/yy |

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

| Parameter | Description | Example |
|---|---|---|
| nls_language | Specifies the ISO language code. The codes are defined by ISO-639.<br>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<br>Default value: en<br>**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.<br>Possible value: An alphanumeric string<br>Default value: None | partition=germany |
| service_class | Specifies the entry class for the Windows service.<br>Possible value: oracle/oai/agent/service/AgentService<br>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.<br>Possible value: A valid PATH setting<br>Default value: None<br>This parameter is only for Microsoft Windows. | service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip |
| service_jdk_dll | Specifies the Dynamic Link Library (DLL) that the adapter JVM should use.<br>Possible value: A valid jvm.dll<br>Default value: jvm.dll<br>This parameter is only for Microsoft Windows. | service_jdk_dll=jvm.dll |
| service_jdk_version | Specifies the JDK version that the adapter Java VM should use.<br>Possible value: A valid JDK version number<br>Default value: 1.4<br>This parameter is only for Microsoft Windows. | service_jdk_version=1.4 |
| service_max_heap_size | Specifies the maximum heap size for the adapter JVM.<br>Possible value: A valid JVM heap size<br>Default value: 536870912<br>This parameter is only for Microsoft Windows. | service_max_heap_size=536870912 |
| service_max_java_stack_size | Specifies the maximum size to which the JVM stack can grow.<br>Possible value: A valid JVM maximum stack size<br>Default value: Default value for the JVM<br>This parameter is only for Microsoft Windows. | service_max_java_stack_size=409600 |

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

| Parameter | Description | Example |
|---|---|---|
| service_max_native_stack_size | Specifies the maximum size the JVM native stack can grow.<br>Possible value: The valid JVM maximum native stack size<br>Default value: Default value for the JVM<br>This parameter is only for Microsoft Windows. | service_max_native_size=131072 |
| service_min_heap_size | Specifies the minimum heap size for the adapter JVM.<br>Possible value: The valid JVM heap size<br>Default value: 536870912<br>This parameter is only for Microsoft Windows. | service_min_heap_size=536870912 |
| service_num_vm_args | Specifies the number of service_vm_arg*number* parameters specified in JVM.<br>Possible value: The number of service_vm_arg*number* parameters<br>Default value: None<br>This parameter is only for Microsoft Windows. | 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.<br>Possible value: The valid PATH environment variable setting<br>Default value: None<br>This parameter is only for Microsoft Windows. | 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.<br>Possible value: A valid JVM arguments<br>Default value: None<br>This parameter is only for Microsoft Windows. | service_vm_arg1=java.compiler=NONE<br>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 |

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

| Letter | Description | Example |
|---|---|---|
| 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` |
| 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.<br>Possible value: `oracle.oai.agent.adapter.technology.TechBridge`<br>Default value: None | `bridge_class=oracle.oai.agent.adapter.technology.TechBridge` |
| `http.receiver.customized_class` | Specifies the class name for customizing the HTTP response.<br>Default value: None | `http.receiver.customized_class=MyBanner` |
| `http.receiver.customizer_class` | Specifies the class name for customizing the HTTP sender.<br>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.<br>Default value: `oai` | `http.receiver.instance_name=oai` |
| `http.receiver.registry_port` | Specifies the RMI port used by the HTTP receiver.<br>Default value: `9901` | `http.receiver.registry_port=9901` |

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

| Parameter | Description | Example |
|---|---|---|
| `http.reqreply.mode` | Specifies the type of mode, synchronous or asynchronous, that can be set for the request reply messaging paradigm.<br>Possible values: `sync`, `async`<br>If the value is async, the reply will be sent to the send endpoint defined by `ota.send.endpoint`.<br>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.<br>Default value: `60s` | `http.reqreply.syncmode.timeout=6000` |
| `http.sender.authtype` | Specifies if authentication is needed.<br>Possible value: `basic` or `digest`.<br>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:<br>`SSL_RSA_WITH_3DES_EDE_CBC_SHA`<br>`SSL_RSA_WITH_RC4_128_SHA`<br>`SSL_RSA_WITH_RC4_128_MD5`<br>`SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`<br>`SSL_DH_anon_WITH_RC4_128_MD5`<br>`SSL_DH_anon_WITH_DES_CBC_SHA`<br>`SSL_RSA_WITH_DES_CBC_SHA`<br>`SSL_RSA_EXPORT_WITH_RC4_40_MD5`<br>`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`<br>`SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`<br>`SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`<br>`SSL_RSA_WITH_NULL_SHA`<br>`SSL_RSA_WITH_NULL_MD5`<br>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.<br>Default value: `oracle.oai.agent.adapter.technology.HTTPDefaultSenderCustomizer` | `http.sender.customizer_class=MyHTTPSenderCustomizer` |

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

| Parameter | Description | Example |
|---|---|---|
| `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`.<br>Default value: None<br>**See Also:** Section A.7, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions" for instructions on how to modify and retrieve the password | `http.sender.password=httpuser` |
| `http.sender.proxy_host` | Specifies the proxy hostname.<br>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.<br>Default value: None | `http.sender.proxy_port=80` |
| `http.sender.realm` | Specifies the realm for the authentication scheme.<br>Default value: None | `http.sender.realm=ipt` |
| `http.sender.timeout` | Specifies the time out for an HTTP connection. The unit is milliseconds.<br>Default value: `60000` milliseconds (60 seconds) | `http.sender.timeout=10000` |
| `http.sender.username` | Specifies the authentication user name.<br>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.<br>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`.<br>Default value: None<br>**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` |

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

| Parameter | Description | Example |
|---|---|---|
| ota.receive.endpoint | Specifies the URL of the receiving application. The URL is of the form:<br>http[s]://hostname:port/path<br>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:<br>http[s]://hostname:port/path<br>Default value: None | ota.send.endpoint=<br>http://site.com:8888/servlet/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.<br>Default value: XML | ota.type=XML |

## 2.4 Uninstalling the HTTP Adapter

To uninstall the HTTP adapter, perform the following:

1. Navigate to the *MiddleTier*\opmn\bin directory.
2. Run the following command to check the adapter status.

   ```
   opmnctl status
   ```

3. If the HTTP adapter instance that you want to remove is running, stop it by using the the following command:

   ```
   opmnctl stopproc ias-component="InterConnect" process-type="HTTPApp"
   ```

   where HTTPApp is the name of the HTTP adapter instance.

4. Navigate to the *MiddleTier*\bin directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

5. Carefully, remove the adapter process-type entry from the opmn.xml file located in the *MiddleTier*\opmn\conf directory. For example, to remove an HTTP adapter instance myHTTPApp1, delete the following information specific to the adapter instance:

   ```
   <process-type id="myHTTPApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/myHTTPApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myHTTPApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
                       <data id="class-name"
   ```

```
                    value="oracle.oai.agent.proxy.ShutdownAgent"/>
                  <data id="application-parameters"
                    value="persistence/Agent.ior"/>
                </category>
              </module-data>
            </process-set>
</process-type>
```

6. Save the `opmn.xml` file.
7. Navigate to the *`MiddleTier`*`\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

8. Navigate to the *`ORACLE_HOME`*`\integration\interconnect\adapters` directory and delete the folder that was created for the removed adapter instance.
9. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

# 3

# Design-Time and Run-Time Concepts

This chapter describes the design-time and run-time 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) or XML Schema Definition (XSD) 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 or XSD 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 Run-Time Concepts

This section describes the key run-time 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 acknowledgment message, whereas in the asynchronous request/reply mode, the adapter will send the acknowledgment 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 in `http.receiver.*` format.

> **Note:** The adapter subscribing to an event should be started before any other adapter can publish that event. If you publish an event before starting the subscribing adapter, then the event would not be delivered to the subscribing adapter.

**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 comprises of a HTTP bridge and a run-time 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 in `http.sender.*` format.

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 box. After associating an endpoint and an 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 enables 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 1-2
- 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 by 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<br>Possible values are:<br>`0` for publish<br>`1` for request<br>`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 The ReceiverCustomizer Interface

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

- To customize the `TransportMessage` object itself, use the `customizeTransportMessage()` method. This method is called before the before the adapter processes the `TransportMessage` object.
- To modify the message itself, 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 |
|---|---|
| `customizeTransportMessage();` | Enables you to customize the transport message received by the adapter. It uses the following parameters:<br>agent: Log a message<br>receiverType: Information on the type of adapter<br>transportMessage: Customize the transport message received by the adapter |
| `createReplyMessage();` | Creates a reply message based on the status of message received. It contains the following parameters:<br>`agent`: Log a message.<br>`status`: Status of the message process. If the value is `TransportResponse.TRANSPORT_ACK`, then the message has been processed successfully. If the value is `TransportResponse.TRANSPORT_ERROR`, then the message has not been processed successfully.<br>`receivedTransportMessage`: Transport message received by the adapter. This parameter is used to transport headers in the transport message to create a meaningful HTTP message.<br>`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.<br>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 can select 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:<br>`in`: Contains the message. It is an Inputstream object.<br>It throws an exception of type `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:<br>`body`: body of the message<br>It throws an exception of type `TransportException`. |
| `public void setTransportHeader(String name, String value);` | Set a transport specific header. |

## 3.3.2 The 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 The 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);

}
```

#### The 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 The 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.
  2. Select the ***OracleHomeOracleASIntegrationInterConnectAdapter-Application*** service.
  3. Start the service based on the operating system.

---

**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 `log.xml` files. The files are located in the time-stamped 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.

2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
3. Stop the service.

You can verify the stop status of the HTTP adapter by viewing the `log.xml` files. These files are located in the time-stamped 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.



## A.1 How do I know whether the HTTP adapter has started properly?

View the `log.xml` file located in the time-stamped 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 2-3, and *timestamp_in_milliseconds* is the directory. If no exceptions are listed, then the adapter has started properly.

## A.2 The HTTP adapter did not start properly. What is wrong?

View the exceptions in the adapter log file (`log.xml`). 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

## A.3 The HTTP adapter is not starting. What could be the reason?

If you are starting the adapter for the first time or if you have set the `agent_metadata_cache parameter` to `none` or `demand`, then check if the repository is up and running. In addition, check if the hub database is up and running.

One reason can be that Oracle Wallet does not contain the password information corresponding to your application name. For example, during installation you defined the application name as `myHTTPApp`. Later, you changed the application name in iStudio to `HTTPApp`. In such case, you need to specify the password corresponding to the new application name `HTTPApp` in the Oracle Wallet. You can create password by using the `oraclewallet` command.

**See Also:** Section A.7, "How do I secure my passwords?"

## A.4 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.

## A.5 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 2-13 for parameter information

## A.6 HTTP adapter does not publish or subscribe to messages

The HTTP adapter is not publishing or subscribing to the messages.

#### Problem 1

The Transport properties of the HTTP adapter might not be correct.

#### Solution 1

Specify 2 as the value of the log level parameter `agent_log_level` in the `adapter.ini` file of the HTTP adapter. Restart the adapter and try to publish a message or subscribe to a message. You can see a log starting with `TransportProperties.TransportProperties()`. This log contains all the transport properties of the HTTP adapter in the form of name value pairs. Ensure that the values of these properties are correct.

#### Problem 2

You might not have created a publish or subscribe event in iStudio for the HTTP adapter that corresponds to the message structure. The HTTP adapter is not able to match the message data to any of the events defined during design time and therefore cannot publish or subscribe the message.

#### Solution 2

Create a publish or subscribe event for the HTTP adapter in iStudio corresponding to the message structure.

#### Problem 3

The transport servlet is not deployed properly.

#### Solution 3

Ensure that the transport servlet that receives the messages over HTTP and directs these messages to the HTTP receiver through RMI is deployed properly. In addition, you should never change the `http.receiver.instance_name` parameter in the `adapter.ini` file after the adapter installation.

#### Problem 4

The send endpoint defined by the `ota.send.endpoint` parameter in the `adapter.ini` file or in iStudio is incorrect.

#### Solution 4

Ensure that the send endpoint defined in the `adapter.ini` file or iStudio is correct. If the send endpoint expects the message in some particular form, then you can customize the outgoing messages using the `SenderCustomizer` parameter.

## A.7 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
```

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.

The number of entries is dependent on the type of adapter. For example, DB Adapter needs two entries whereas AQ Adapter needs only one entry. The following table lists the entries that will be created for each adapter:

| Adapter | Entry In Oracle Wallet |
|---|---|
| AQ | *ApplicationName*/aq_bridge_password |
| HTTP | *ApplicationName*/http.sender.password |
| HTTP | *ApplicationName*/sender.wallet_password |
| SMTP | *ApplicationName*/smtp.receiver.password |
| MQ | *ApplicationName*/mq.default.password |
| FTP | *ApplicationName*/file.sender.password |
| FTP | *ApplicationName*/file.receiver.password |
| DB | *ApplicationName*/db_bridge_schema1_password |
| DB | *ApplicationName*/db_bridge_schema1_writer_password |

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
```

## A.8 How can I deliver a message to a specific partition of the publishing adapter?

**Scenario**: HTTP adapter has two partitions `PAR1` and `PAR2`. You want to deliver event `create_customer` to partition `PAR1` and `add_customer` event to partition `PAR2`.

Perform the following tasks:

1. Assign event `create_customer` to partition `PAR1` and event `add_customer` to partition `PAR2`.
2. Deploy two transportServlets in the OC4j container. For example `TRANSPORTSERVELET1` and `TRANSPORTSERVELET2`.
3. Configure the adapter with partition name `PAR1` to receive messages from `TRANSPORTSERVLET1` and adapter with partition name `PAR2` to to receive messages from `TRANSPORTSERVLET2`.
4. Publish the message for event `create_customer` to `TRANSPORTSERVLET1` and the event `add_customer` to `TRANSPORTSERVLET2`.

# 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 2-12 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

## A



## B



## C



## D



## E

## H



## I



## L

## M



## O



## P



## R



## S



## T



## U



## W



## X