Oracle® Application Server InterConnect

Adapter for FTP Installation and User's Guide

10*g* (9.0.4)

Part No. B10412-01

September 2003

ORACLE®

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

Part No. B10412-01

Contributing Authors: Ching Chung and Min Yao

Contents

Send Us Your Comments

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

Part No. B10412-01

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

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

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

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

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

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

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

Preface

This preface contains these topics:

Audience

Oracle Application Server InterConnect Adapter for FTP Installation and User's Guide is the primary source of introduction, installation, and usage information for the File Transfer Protocol (FTP) adapter.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

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

Organization

This document contains:

Chapter 1, "Introduction"

This chapter describes the Oracle Application Server InterConnect Adapter for FTP (FTP adapter) and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the FTP adapter.

Chapter 3, "Design Time and Runtime Concepts"

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

Chapter 4, "Frequently Asked Questions"

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

Appendix A, "Sample Adapter.ini File"

This appendix provides a sample `adapter.ini` file for the FTP adapter.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Application Server InterConnect User's Guide*
- *Oracle Application Server InterConnect Installation Guide*
- *Oracle Application Server InterConnect Adapter Configuration Editor User's Guide*

Printed documentation is available for sale in the Oracle Store at

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

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

`http://otn.oracle.com/membership/`

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

`http://otn.oracle.com/documentation/`

Conventions

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

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

1

Introduction

This chapter discusses Oracle Application Server InterConnect Adapter for FTP (FTP adapter) and the hardware and software requirements. This chapter provides an overview of the FTP adapter.

FTP Adapter Overview

The FTP adapter enables an Oracle FTP Application to be integrated with other applications using Oracle Application Server InterConnect (OracleAS InterConnect). This adapter is useful in all enterprise application integration scenarios involving the FTP transport protocol or local file system. The FTP adapter can monitor incoming messages which are in the form of FTP files placed in a remote FTP server or on local file systems. The FTP adapter is also capable of sending messages to remote FTP servers via proxy host. The payload for this adapter can be XML data or D3L data.

The following two diagrams show the data flow of incoming messages, from remote FTP server/local file system to OracleAS InterConnect, and the outgoing messages from OracleAS InterConnect to the remote FTP Server.

See Also: *Oracle Application Server InterConnect User's Guide*

Figure 1–1 Incoming Messages Diagram

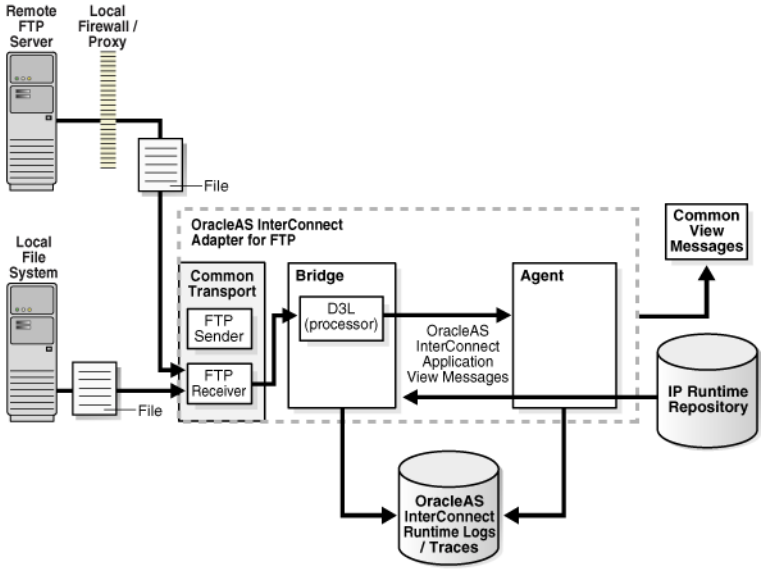
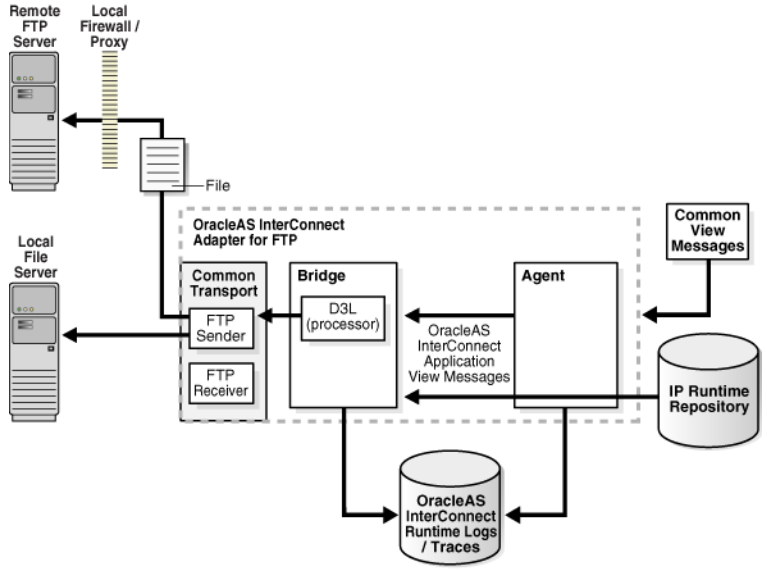


Figure 1–2 Outgoing Messages Diagram



Hardware Requirements

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

Table 1–1 Hardware Requirements

Hardware	Windows	UNIX
Memory	128 MB	128 MB
Service Pack	NT 4.0–Service Pack 6 or above 2000–Service Pack 1 or above	N/A
Disk Space	500 MB	500 MB

Software Requirements

The following are software requirements for the FTP adapter:

- Operating System Requirements
- JRE Requirements

Operating System Requirements

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

Table 1–2 Operating System Requirements

Operating System	Version
Windows NT	Version 4.0 with Service Pack 6 or above
Windows 2000	With Service Pack 1 or above
IBM AIX 5L	5.1 and 5.2 (64 bit)
HP Tru64	5.1a and 5.1b (64 bit)
HP-UX	11.0 and 11.11 (64 bit)
RedHat Linux	Advanced Server 2.1
Sun SPARC Solaris	8 and 9 (32 bit)

JRE Requirements

OracleAS InterConnect uses JRE 1.4.1 which is installed with its components.

Known Limitations

The FTP adapter has the following limitations:

- Does not support secure FTP server.
- Does not support file-filtering features.
- Can only receive from a single FTP or file endpoint.
- The sending endpoint and receiving endpoint are restricted to FTP and file endpoints.

2

Installation and Configuration

This chapter describes how to install and configure the FTP adapter.

This chapter discusses these topics:

- Installing the FTP Adapter
- FTP Adapter Configuration
- Starting the FTP Adapter
- Stopping the FTP Adapter

Installing the FTP Adapter

This section contains these topics:

Preinstallation Tasks

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

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

Consult the following guides before proceeding with FTP adapter installation:

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

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

Installation Tasks

To install the FTP adapter:

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

   Consider the following scenarios:

   - If installing the FTP Adapter in an independent Oracle home, make sure that the OracleAS InterConnect Hub has been installed, not necessarily in the same Oracle home. Continue to step 2.

- If installing the FTP adapter in an existing Oracle home, make sure that it is a home directory to one of the OracleAS InterConnect component. Continue to step 3.

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

---

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

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

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

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

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

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

4. Click **Next**.

   The Oracle Application Server InterConnect for FTP Adapter usage page displays.

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

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

6. Enter the following information in the OracleAS InterConnect FTP Adapter Configuration - Configure sending endpoint information page:
   - Username—The username for the FTP server.
   - Password—The password for the FTP server.
   - FTP Mode—The mode of access used to send information to the specified URL. Select either binary or ascii.
   - URL—The URL to be used for sending information. Enter one of the following:
     * For sending to an FTP server—`ftp://<host name>/<path>`
     * For sending to a local file system—`file://localhost/<path>`

     Note: If the sender endpoint is a local file system, the user name, password, and file type are not required.

7. Click **Next**.

   The installation page that displays next is based on the selection you made in Step 5:

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

8. Enter the following information in the OracleAS InterConnect FTP Adapter Configuration - Configure receiving endpoint information page:
   - Username—The username account of the FTP server from which the Oracle Application Server InterConnect Hub receives messages

- Password—The password for the username account
- FTP Mode—The mode of access used to receive information from the specified URL. Select either binary or ascii.
- URL—The FTP URL to be used for receiving information. Enter one of the following:
  * For sending to an FTP server—`ftp://<host name>/<path>`
  * For sending to a local file system—`file://localhost/<path>`

Note: If the sender endpoint is a local file system, the user name, password, and file type are not required.

Caution: For testing purposes, do not specify a personal FTP account or personal file directories as the receiving endpoint. During runtime, the FTP adapter connects to the FTP server or accesses the file system and removes the files in the directory specified by the receiving endpoint after processing. Oracle Corporation recommends that you create a dedicated FTP account or user account (if the local file system is used for the receiving endpoint) for testing and deploying this adapter.

9. Click **Next**.
10. Complete any other fields for other components selected for installation, such as other adapters.

    When finished, the Summary page displays.
11. Click **Install** to install the FTP adapter. The adapter is installed in the following directory:

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

    You have defined the value of `Application` in Step 3.
12. Click **Exit** at the End of Installation page to exit the FTP adapter installation.

Postinstallation Tasks

FTP adapter installation creates an `adapter.ini` file that consists of configuration parameters read by the FTP adapter at startup. These configuration parameter settings are appropriate for most FTP application environments. You can customize some `adapter.ini` file parameter settings for the FTP application after installation. See the following sections:

- Customizing the Payload Data Type
- Customizing the Sending Endpoints
- Customizing the Receiving Endpoints

See Also:

- Table 2–1 on page 2-10 for the location of the `adapter.ini` file
- Table 2–7 on page 2-21 for `adapter.ini` file parameter setting information specific to the FTP adapter

Customizing the Payload Data Type

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

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

   ```
   ota.type=D3L
   ```

2. Copy the D3L XML files associated with the FTP 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 FTP application. For example:

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

See Also: ota.type and ota.d3ls parameter descriptions on page 2-21

Customizing the Sending Endpoints

If you want to customize the behavior of the sending endpoints (destinations) for messages, edit the following parameters in the `adapter.ini` file. These parameters are not automatically set to default values during installation.

1. Change the sender endpoint by editing the `ota.send.endpoint` parameter or leave it blank if it acts only as a receiver. For example:
   - For a remote file system—`ota.send.endpoint=ftp://foo.com/test`
   - For a local file system—`ota.send.endpoint=file://localhost/test`

   If the endpoint is a local file system, leave the following parameters blank:
   - `file.sender.file_type`
   - `file.sender.password`
   - `file.sender.proxy_host`
   - `file.sender.proxy_port`
2. Set the `file.sender.file_type` parameter to the file type used in FTP. For example:

   ```
   file.sender.file_type=BINARY
   ```
3. Update the `file.sender.user` and `file.sender.password` parameters with the information of the FTP account that serves as the sending endpoint.
4. If a proxy host is needed, enter the appropriate values for the `file.sender.proxy_host` and `file.sender.proxy_port` parameters.
5. Set the `file.sender.staging_dir` parameter. This parameter prevents partial files picked up by external applications.

   See Also: FTP Sender in Chapter 3 for details
6. Set the `file.sender.file_name_rule` parameter. This parameter controls how the adapter generates the file name.

   See Also: FTP Sender in Chapter 3 for details
7. If you need to modify the contents of an outgoing message before it is sent by the transport layer, you can customize it by implementing the `FileSenderCustomizer` Interface. You need to set the `file.sender.customize_class` to the name of the customizing class.

Customizing the Receiving Endpoints

If you want to customize the behavior of the receiving FTP or file endpoints for messages, edit the following parameters in the `adapter.ini` file.

1. Change the receiver endpoint by editing the ota`.receive.endpoint` or leave it blank if the adapter only acts as a sender. For example:
   - For a remote file system—`ota.receive.endpoint=ftp://foo.com/test`
   - For a local file system—`ota.receive.endpoint=ftp://localhost/test`

   Note: If the endpoint is a local file system leave the following parameter blank:

   `file.receiver.file_type`, `file.receiver.password`, `file.receiver.proxy_host`, and `file.receiver.proxy_port`

   Warning: Do not use the ota.receive.endpoint to a personal file directory as the files in that directory will be consumed and deleted by the FTP adapter after processing.

2. Set the `file.receiver.file_type` parameter to the file type used in FTP. For example:

   ```
   file.receiver.file_type=BINARY
   ```

3. Update the `file.receiver.user` and `file.receiver.password` parameters with the information of the FTP account that serves as the receiving endpoint.
4. If a proxy host is needed, enter the appropriate values for the `file.receiver.proxy_host and file.receiver.proxy_port` parameters.
5. Set the `file.exception.exception_dir` to a local file system directory that stores files and cannot be processed successfully. For example:

   ```
   file.receiver.exception_dir=/tmp/error
   ```

6. Set the `file.receiver.polling_interval` parameter to the time interval in milliseconds during which to poll the FTP server or local file system. For example:

   ```
   file.receiver.polling_interval=2000
   ```

7. Set the `file.receiver.max_msgs_retrieved` parameter to the maximum number of messages to retrieve in polling a session. For example:

   ```
   file.receiver.max_msgs_retrieved=10
   ```

8. If you need to modify the contents of an incoming message before it is processed by the bridge, for example, to remove an extra line in a file, you can customize it by implementing the `FileSenderCustomizer` interface. You need to set the `file.receiver.customize_class` to the name of the customizing class.

FTP Adapter Configuration

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

Table 2–1 FTP Adapter Directory

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

Table 2–2 FTP Executable Files

File	Description
`start.bat` (Windows) `start` (UNIX)	Takes no parameters; starts the adapter
`stop.bat` (Windows) `stop` (UNIX)	Takes no parameters; stops the adapter
`ignoreerrors.bat` (Windows) `ignoreerrors` (UNIX)	If an argument is specified, then the given error code is ignored: `ignoreerrors` *errorCodeToBeIgnored* If no argument is specified, then all error codes specified in the `ErrorCodes.ini` file are ignored: `ignoreerrors`

See Also: "FTP Error Code" on page 3-13 for a list of error codes

Table 2–3 FTP Configuration Files

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

See Also: Appendix A, "Sample Adapter.ini File"

Table 2–4 FTP Directories

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

Using the Application Parameter

Adapters do not have integration logic. The FTP adapter has a generic transformation engine that processes metadata from the repository as runtime instructions to do transformations. The application defines for an adapter what its capabilities are. For example, it can define what messages it can publish, what messages it can subscribe to, and what are the transformations to perform. The application parameter allows the adapter to become smart in the context of the application to which it is connected. It allows the adapter to retrieve from the repository only that metadata that is relevant to the application. The application parameter must match the corresponding application that will be defined in iStudio under the Applications folder.

If you are using pre-packaged metadata, after importing the pre-packaged metadata into the repository, start up iStudio to find the corresponding application (under the Applications folder in iStudio) to use as the application for the adapter you are installing (unless the package you are using provides directions for what the application should be).

Ini File Settings

The following are `.ini` files used in configuring the FTP adapter.

- Hub.ini Parameters
- Adapter.ini File

Hub.ini Parameters

The FTP adapter connects to the hub database using parameters from the `hub.ini` file located in the hub directory. The following table lists the parameter name, a description for each parameter, the possible and default values, and an example.

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

Real Application Clusters-specific Hub.ini Parameters

When a hub is installed on a Real Application Clusters (RAC) database, parameters listed in Table 2–5 represent information on additional nodes used for connection and configuration. These parameters are added on top of the default parameters which represent the primary node. In Table 2–5, *x* represent the node number, which varies between 2 and the number of nodes. For example, if the Real Application Clusters setup contains 4 nodes, *x* can take a value between 2 and 4.

Table 2–5 Real Application Clusters-specific hub.ini Parameters

Parameter	Description	Example
hub_num_nodes	Number of nodes in Real Application Clusters.	hub_num_nodes=4
hub_host*x*	The host where the Real Application Clusters database is installed.	hub_host2=dsunram13
hub_instance*x*	The instance on the respective node.	hub_instance2=orcl2
hub_port*x*	The port on which the listener is listening.	hub_port2=1521

Adapter.ini File

The agent component of the FTP adapter reads the `adapter.ini` file at runtime to access FTP adapter parameter configuration information. Table 2–6 lists the parameter name, a description for each parameter, the possible and default values, and an example.

Table 2–6 Agent Connection Parameters

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

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`agent_reply_subscriber_name`	Specifies the subscriber name used when multiple adapter instances for the given application with the given partition are used. This parameter is optional if there is only one instance running. The possible value is the application name (*`parameter`*`:`*`application`*) concatenated with the instance number (*`parameter`*`:`*`instance_number`*). There is no default value.	If `application=ftpapp` and `instance_number=2`, then `agent_reply_subscriber_name=ftpapp2`
`agent_reply_message_selector`	Used only if there are multiple adapter instances for the given application with the given partition. The possible value is a string built using the concatenated application name (*`parameter`*`:`*`application`*) with the instance number (*`parameter`*`:`*`instance_number`*). There is no default value.	If `application=ftpapp` and `instance_number=2`, then `agent_reply_message_selector=recipient_list like'%,ftpapp2,%'`
`agent_tracking_enabled`	Specifies if message tracking is enabled. Set this parameter to `false` to turn off all tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. Possible values are `true` or `false`. The default value is `true`.	`agent_tracking_enabled=true`
`agent_throughput_measurement_enabled`	Specifies if throughput measurement is enabled. Set this parameter to `true` to turn on all throughput measurements. The default value is `true.`	`agent_throughput_measurement_enabled=true`
`agent_use_custom_hub_dtd`	Specifies whether to use a custom document type definition (DTD) for the common view message when handing it to the hub (the repository in which metadata is stored). By default, adapters use an OracleAS InterConnect-specific DTD for all messages sent to the hub, as other OracleAS InterConnect adapters retrieve the messages from the hub and know how to interpret them. Set this parameter to `true` if you want the adapter to use the DTD imported for every message of the common view. Only set this parameter to `true` if an OracleAS InterConnect adapter is not receiving the messages from the hub. There is no default value.	`agent_use_custom_hub_dtd=false`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`agent_metadata_caching`	Specifies the metadata caching algorithm. Possible values are: ■ `startup`—Cache everything at startup. This may take a while if there are many metadata in the repository. ■ `demand`—Cache metadata as it is used. ■ `none`—No caching. This slows down performance. The default value is `demand`.	`agent_metadata_caching=demand`
`agent_dvm_table_caching`	Specifies the domain value mapping (DVM) table caching algorithm. Possible values are: ■ `startup`—Cache all DVM tables at startup. This may take a while if there are many tables in the repository. ■ `demand`—Cache tables as they are used. ■ `none`—No caching. This slows down performance. The default value is `demand`.	`agent_dvm_table_caching=demand`
`agent_lookup_table_caching`	Specifies the lookup table caching algorithm. Possible values are: ■ `startup`—Cache all lookup tables at startup. This may take a while if there are many tables in the repository. ■ `demand`—Cache tables as they are used. ■ `none`—No caching. This slows down performance. The default value is `demand`.	`agent_lookup_table_caching=demand`
`agent_delete_file_cache_at_startup`	With any of the agent caching methods enabled, metadata from the repository is cached locally on the file system. Set this parameter to `true` to delete all cached metadata on startup. Possible values are `true` or `false`. The default value is `false`. **Note:** After changing metadata or DVM tables for this adapter in iStudio, you must delete the cache to guarantee access to the new metadata or table information.	`agent_delete_file_cache_at_startup=false`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`agent_max_ao_cache_size`	Specifies the maximum number of application objects' metadata to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_ao_cache_size=200`
`agent_max_co_cache_size`	Specifies the maximum number of common objects' metadata to cache. Possible values are any integer greater than or equal to `1`. The default value is `100`.	`agent_max_co_cache_size=100`
`agent_max_message_metadata_cache_size`	Specifies the maximum number of messages' metadata (publish/subscribe and invoke/implement) to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_message_metadata_cache_size=200`
`agent_max_dvm_table_cache_size`	Specifies the maximum number of DVM tables to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_dvm_table_cache_size=200`
`agent_max_lookup_table_cache_size`	Specifies the maximum number of lookup tables to cache. Possible values are any integer greater than or equal to `1`. The default value is `200`.	`agent_max_lookup_table_cache_size=200`
`agent_max_queue_size`	Specifies the maximum size to which internal OracleAS InterConnect message queues can grow. Possible values are any integer greater than or equal to `1`. The default value is `1000`.	`agent_max_queue_size=1000`
`agent_persistence_queue_size`	Specifies the maximum size to which internal OracleAS InterConnect persistence queues can grow. Possible values are any integer greater than or equal to `1`. The default value is `1000`.	`agent_persistence_queue_size=1000`
`agent_persistence_cleanup_interval`	Specifies how often to run the persistence cleaner thread (in milliseconds). Possible values are any integer greater than or equal to `30000` milliseconds. The default value is `60000`.	`agent_persistence_cleanup_interval=60000`
`agent_persistence_retry_interval`	Specifies how often for the persistence thread to retry when it fails to send an OracleAS InterConnect message. Possible values are any integer greater than or equal to `60000` milliseconds. The default value is `60000`.	`agent_persistence_retry_interval=60000`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`agent_pipeline_to_hub`	Specifies how to turn on or off the pipeline for messages from the Bridge towards the hub. If you set the pipeline to `false`, the file persistence is not used in that direction.	`agent_pipeline_to_hub=false`
`agent_pipeline_from_hub`	Specifies how to turn on or off the pipeline for messages from the hub towards the Bridge. If you set the pipeline to `false`, the file persistence is not used in that direction.	`agent_pipeline_from_hub=false`
`service_path`	Windows only. Specifies the value to which to set the environment variable `PATH`. The `PATH` variable is set to the specified value before forking the Java VM. Typically, all directories containing all necessary DLLs should be listed here. Possible values are the valid path environment variable setting. There is no default value.	`service_path=%JREHOME%\bin;D:\oracle\ora904\bin`
`service_classpath`	Specifies the class path used by the adapter Java VM. If a custom adapter is developed and, as a result, the adapter is to pick up any additional jars, add the jars to the existing set of jars being picked up. Possible values are the valid class path. There is no default value.	`service_classpath=D:\oracle\ora904904\oai\904\lib\oai.jar;%JREHOME%\lib\rt.jar;D:\oracle\ora904\jdbc\classes12.zip`
`service_class`	Specifies the entry class for the Windows service. A possible value is `oracle/oai/agent/service/AgentService`. There is no default value.	`service_class=oracle/oai/agent/service/AgentService`
`service_max_java_stack_size`	Windows only. Specifies the maximum size to which the Java VM's stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	`service_max_java_stack_size=409600`
`service_max_native_stack_size`	Windows only. Specifies the maximum size to which the Java VM's native stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	`service_max_native_stack_size=131072`
`service_min_heap_size`	Windows only. Specifies the minimum heap size for the adapter Java VM. Possible values are the valid Java VM heap sizes. The default value is the default Java VM heap size.	`service_min_heap_size=536870912`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`service_max_heap_size`	Windows only. Specifies the maximum heap size for the adapter Java VM. Possible values are any valid Java VM heap sizes. The default value is `536870912`.	`service_max_heap_size=536870912`
`service_num_vm_args`	Windows only. Specifies the number of `service_vm_arg`*`number`* parameters specified. Possible values are the number of `service_vm_arg`*`number`* parameters. There is no default value.	`service_num_vm_args=1`
`service_vm_arg`*`number`*	Windows only. Specifies any additional arguments to the Java VM. For example, to get line numbers in any of the stack traces, set `service_vm_arg1=java.compiler=NONE`. If there is a list of arguments to specify, use multiple parameters as shown in the example by incrementing the last digit starting with `1`. Be sure to set `service_num_vm_args` correctly. Possible values are any valid Java VM arguments. There is no default value.	`service_vm_arg1=java.compiler=NONE` `service_vm_arg2=oai.adapter=database`
`corba_port_number`	The CORBA port number on which the adapter CORBA service listens. Generally, this port is allocated dynamically. However, it can be configured to enable access across firewall.	`corba_port_number=14000`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example

`nls_date_format`	Format for date fields expressed as string. The following pattern letters are defined. All other characters from A to Z and from a to z are reserved.
	
	Letter — Date or Time Component — Examples
	`G` — Era designator — `AD`
	`y` — Year — `1996; 96`
	`M` — Month in year — `July; Jul; 07`
	`w` — Week in year — `27`
	`W` — Week in month — `2`
	`D` — Day in year — `189`
	`d` — Day in month — `10`
	`F` — Day of week in month — `Number 2`
	`E` — Day in week — `Tuesday; Tue`
	`a` — A.M./P.M. marker — `P.M.`
	`H` — Hour in day (0-23) — `0`
	`k` — Hour in day (1-24) — `24`
	`K` — Hour in A.M/P.M. (0-11) — `0`
	`h` — Hour in A.M./P.M. (1-12) — `12`
	`m` — Minute in hour — `30`
	`s` — Second in minute — `55`
	`S` — Millisecond — `978`
	`z` — Time zone — `Pacific`
	
	The default date format is `EEE MMM dd HH:mm:ss zzz yyyy`.
	
	Note: This parameter specifies date format. It is applicable for the date format only.
		Date format pattern `dd/MMM/yyyy` can represent 01/01/2003.
		
		`nls_date_format=dd-MMM-yy`
		
		Multiple date format can be specified as `num_nls_formats=2`
		
		`nls_date_format1=dd-MMM-yy`
		
		`nls_date_format2=dd/MMM/yy`

Table 2–6 Agent Connection Parameters

Parameter	Description	Example
`nls_language`	This parameter is a valid ISO Language Code. These lower-case and two-letter codes are defined by ISO-639. You can find a full list of these codes at a Web site, such as, `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt` The default language code is `en`. **Note**: This parameter specifies date format. It is applicable for the date format only.	`nls_language=en`
`nls_country`	This parameter is a valid ISO Country Code. These upper-case and two-letter codes are defined by ISO-3166. You can find a full list of these codes at a Web site, such as, `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html` The default Country code is `US`. **Note**: This parameter specifies date format. It is applicable for the date format only.	`US`
`encoding`	Character encoding for published messages. The adapter uses this parameter to generate encoding information in encoding tag of transformed OracleAS InterConnect message. OracleAS InterConnect represents messages internally as an XML document. The default encoding of the XML document is `UTF-8`. However, this encoding can be configured using this parameter, which is typically used when the OracleAS InterConnect message consists of characters not supported by `UTF-8` and when the `XMLParser` is unable to handle them.	`encoding=JA16SJIS`

FTP Adapter-Specific Parameters

Table 2–7 lists the parameters specific to the FTP adapter. With the exception of the `bridge_class` parameter, all parameters can be edited after installation.

Table 2–7 FTP Adapter-Specific Parameters

Parameter	Description	Example
`bridge_class`	Specifies the entry class for the FTP adapter. A value must be specified and cannot be modified later. A possible value is `oracle.oai.agent.adapter.technology. TechBridge`. There is no default value.	`bridge_ class=oracle.oai.agent.ada pter.technology. TechBridge`
`ota.send.endpoint`	Defines the FTP sending endpoint url. The url is written as follows: `ftp://<host name>/<directory path>` or `file://localhost/<directory path>`. The possible values are `ftp://<host name>/<directory path>`. There is no default value. **Note**: Do not set the `ota.send.endpoint` parameter to the same value set for the `ota.receive.endpoint` parameter unless you perform a loop-back test for the FTP adapter. The files that you send out are used by the receiving end.	`ota.send.endpoint=ftp://ip -sun/private/ ipdev1/test/inbound`

`ota.receive.endpoint`	Defines the FTP receiving endpoint url. The url is written as follows: `ftp://<host name>/<directory path>` or `file://localhost/<directory path>`. The possible values are `ftp://<host name>/<directory path>`. There is no default value.
	
	On Windows platforms, if you have a file system endpoint, you can use the drive letter and directory name as part of the endpoint url.
		`ota.receive.endpoint= ftp://ip-sun/private/ ipdev1/test/inbound`
		
		On Windows:
		
		`ota.receive.endpoint=file: //localhost/c:\test`

`ota.type`	Defines the type of payload this adapter handles. Possible values are XML and D3L. There is no default value.	`ota.type=XML`

Table 2–7 FTP Adapter-Specific Parameters

Parameter	Description	Example
`ota.d3ls`	Specifies the list of data definition description language (D3L) XML files used by this bridge. Each business event handled by the bridge must have its own D3L XML file. Whenever a new D3L XML file is imported in iStudio for use by an application using the SMTP adapter, the parameter must be updated and the SMTP adapter restarted. There is no default value.	`ota.d3ls=person.xml, person1.xml`
`file.sender.user`	The FTP user name for the outbound FTP server. The possible value is a valid FTP user name. There is no default value.	`file.sender.user=joe`
`file.sender.type`	Indicates the file types. The possible values are ASCII or BINARY. The default value is BINARY.	`file.sender.type=ASCII`
`file.sender.proxy_host`	The name of the machine that server as the proxy server for the outbound FTP server. The possible value is any correct host name. There is no default value.	`file.sender.proxy_host=www-proxy.foo.com`
`file.sender.proxy_port`	The port number of the proxy server for the outbound FTP server. The possible value is any valid port number. There is no default value.	`file.sender.proxy_port=80`
`file.sender.staging_dir`	The staging directory name for the file sender. There is no default value.	`file.sender.staging_dir=/private/ipdev1/ftpstaging1`
`file.sender.file_name_rule`	The rule for generating file names used by the file sender. The default value is `%APP%%PART%_%TIME%`	`file.sender.file_name_rule=%APP%_%EVENT%_%TIME%.xml`
`file.sender.customizer_class`	The class name for customization used by the file sender. The default value is: `oracle.oai.agentt.adapter.technology.FileDefaultSenderCustomizer`	`file.sender.customizer_class=MyFileSenderCustomizer`
`file.receiver.user`	The FTP user name for the inbound FTP server. The possible value is any valid FTP user name. There is no default value.	`file.receiver.user=joe`

Table 2–7 FTP Adapter-Specific Parameters

Parameter	Description	Example
`file.receiver.proxy_host`	The name of the machine that servers as the proxy server for the inbound FTP server. The possible value is a correct host name. There is no default value.	`file.receiver.proxy_host=www-proxy.foo.com`
`file.receiver.polling_interval`	Defines the time interval to poll the message source in milliseconds. The default value is `6000`.	`file.receiver.polling_interval = 10000`
`file.receiver.max_msgs_retrieved`	Defines the maximum number of messages to be retrieved in each session. The default value is `30`.	`file.receiver.max_msgs_retrieved = 10`

`file.receiver.exception_dir`	The value of this parameter should be a URL which represents either an FTP directory or a file location. For this release, if an FTP URL can only be specified for the exception directory if the receiving endpoint is also an FTP URL, and furthermore the host name in the URL has to be the same. When a processing exception occurs, the host name, user name, and password of the receiving endpoint will be used to log on to the FTP server to store the messages that are not processed successfully. The user should make sure this directory exists on the FTP server (or the local file system if file URL is used) and is writable by the FTP adapter process.	`file.receiver. execption_dir=ftp://acme.com/private/user/error`
		or
		`file.receiver.exception_dir=file://localhost/private/user/error`

`file.receiver.proxy_port`	The port number of the proxy server for the inbound FTP server. The possible value is a valid port number. There is no default value.	`file.receiver.proxy_port=80`
`file.receiver.customizer_class`	The class name for customization used by the file receiver. The default value is: `oracle.oai.aget.adapter.technology.DefaultReceiverCustomizer`	`file.receiver.customizer_class=MyFileReceiverCustomizer`

Starting the FTP Adapter

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

ORACLE_HOME/oai/9.0.4/adapters/*Application*

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

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

1. Access the Services window from the Start menu:

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

   The Services window appears.

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

On...	Choose...
Windows NT	Choose Start.
Windows 2000	Right click the service and choose **Start** from the menu.

Sample Log File of Successfully Started Advanced Queuing Adapter

The following file displays an FTP adapter that was started successfully:

```
D:\oracle\ora904\oai\9.0.4\adapters\ftpapp>D:\oracle\ora904\oai\9.0.4•in\JavaSer
vice.exe -debug "Oracle OAI Adapter 9.0.4 - ftpapp"
D:\oracle\ora9041\oai\9.0.4\adapters\ftpapp adapter.ini
The Adapter service is starting..
Registering your application (FTPAPP)..
Initializing the Bridge oracle.oai.agent.adapter.technology.TechBridge..
Starting the Bridge oracle.oai.agent.adapter.technology.TechBridge..
Service started successfully.
```

Stopping the FTP Adapter

On UNIX, stop the FTP adapter using the `stop` script in the following directory:

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

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

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

1. Access the Services window from the Start menu:

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

   The Services window appears.

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

On...	Choose...
Windows NT	Choose Stop.
Windows 2000	Right click the service and choose **Stop** from the menu.

You may verify the stop status by viewing the `oailog.txt` files in the appropriate time stamped subdirectory of the `log` directory within the adapter directory.

3

Design Time and Runtime Concepts

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

FTP Adapter Design Time Concepts

The FTP adapter can handle XML and D3L structured payloads. For example:

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

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

XML Payload

Users can import a DTD in iStudio which governs how the FTP adapter will parse a received XML document into a OracleAS InterConnect application view event. In addition, the DTD describes how an inbound application view message will be converted into an XML document. Use the `XML` message type when defining a new integration point in any of the event wizards.

> **Note:** The `oracle.oai.agent.adapter.technology.type` parameter in the `adapter.ini` file should be set to `XML` instead of `D3L`.

When the adapter operates in the XML payload mode, no transformations are performed on the messages between, native view and application view, that are sent or received through the FTP adapter. This is apart from the implied straight ASCII <-> Java object conversion or parsing. XSLT transformations should be performed either before sending an XML document to OracleAS InterConnect, or after receiving one from OracleAS InterConnect.

D3L Payload

The FTP adapter supports both XML and D3L data types. It converts and transforms application view messages to native format and back again.

When an application based on the FTP adapter is created in iStudio, the D3L message type and data type can be used. When these options are selected, messages received or sent by the FTP adapter must adhere to the fixed byte level layout defined in an D3L XML file.

When creating a common view in iStudio, the D3L import option can also be used.

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

FTP Adapter Runtime Concepts

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

How the FTP Adapter Works

This section describes the following components of the FTP adapter:

- FTP Receiver—From remote FTP server or local file system to OracleAS InterConnect functionality.
- FTP Sender—From OracleAS InterConnect to remote FTP server functionality.

FTP Receiver

On the receiving side, the FTP adapter can receive messages from a single receiving endpoint which is either a remote FTP server or a local file system. The receiving endpoint is in the following form:

- For an FTP protocol: `ftp://<host name>/<directory path>`
- For a local file system: `file://localhost/<directory path>`

Based on header name and value information, the FTP bridge uses D3L/XML to parse from native-format into an OracleAS InterConnect message object and generates an application view event. The agent converts the application view event into a common view event and hands it off to OracleAS InterConnect for further routing and processing. Once the message is successfully handled off to OracleAS InterConnect, the corresponding FTP file on the remote FTP server or local file system will be removed. If an error occurs, an exception directory on the remote FTP server or local file system can be specified for storing the unsuccessfully processed files. If no exception directory is specified, the file will be discarded.

Note: The value of the exception directory should be a URL which represents either an FTP directory or a file location. For this release, an FTP URL can only be specified for the exception directory if the receiving endpoint is also an FTP URL, and furthermore the host name in the URL has to be the same. When a processing exception occurs, the host name, user name, and password of the receiving endpoint will be used to log on to the FTP server to store the messages that are not processed successfully. The user should make sure this directory exists on the FTP server (or the local file system if file URL is used) and is writable by the FTP adapter process.

If the `file.receiver.exception_dir` is set, the unsuccessfully processed files will be moved to an exception directory. The archive of exception files of the FTP adapter is appended with a time stamp `<original filename>_<timestamp>`. This is to avoid the two incoming files having the same file name.

The properties for the FTP receiver are found in the `adapter.ini` file as `file.receiver.*`.

FTP Sender

The FTP adapter supports sending to multiple FTP endpoints. This feature provides flexibility for sending messages to different remote FTP servers. An endpoint can be associated with a subscribing event in the iStudio by adding transport properties such as FTP endpoint, FTP user name, and password for this endpoint as metadata or modified fields for the event. Once the association of the endpoint and event is established, the message from the subscribing event is sent out to the FTP endpoint.

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

For example, the following metadata are associated to an event called SendOrder which sends an order to a FTP server at foo.com in the `/private/user/test/directory`.

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

Parameter	Description
`ota.endpoint=sendOrderAppEP`	Specifies a unique endpoint name.
`ota.send.endpoint=ftp://foo.com/private/user/test`	Defines the FTP endpoint.
`file.sender.user=joe`	Defines the FTP user credentials.
`file.sender.password=welcome`	Defines the FTP user credentials.

See Also: *Oracle Application Server InterConnect User's Guide*

The FTP adapter is comprised of the FTP bridge and the runtime agent. When the agent has a message to send to an endpoint, the bridge is notified. The bridge then uses D3L or XML to perform the conversion of the common view object to the native format. The native format message is then sent through the FTP transport layer to a FTP endpoint. The FTP endpoint is written as follows:

```
ftp://<host name>/<directory path>
```

The multi-endpoint feature provides the flexibility to send messages to different FTP servers. The file name at the destination site is automatically generated by the adapter and is in the following form:

```
<ftp adapter name><partition>-<time stamp>
```

The user can specify a rule for generating the file name when the FTP adapter sends a file to a directory or FTP server. To use this feature, the user has to add the parameter, `file.sender.file_name_rule`, in the `adapter.ini` file. The adapter recognizes the following tokens:

%APP%—application name

%PART%—partition number

%BO%—business object name

%TYPE%—message type

%EVENT%—corresponding event name

%TIME%—time stamp

%MV%—message version

For example, `file.sender.file_name_rule=%APP%_%EVENT%_%TIME%.xml`. This rule instructs the adapter to generate files with the following pattern:

```
<your app name>_<event name>_<current time stamp>.xml
```

If the above rule does not serve the user's needs, the user can write his own customization rule by implementing the following interface:

```
oracle.oai.adapter.agent.technology.FileSenderCustomizer interface
```

To avoid external applications picking up partial files, files that are not completely transferred, the user can specify a staging directory. The parameter `file.sender.staging_dir` in the `adapter.ini` file can be set. This parameter should only contain the directory name, for example, `file.sender.staging_dir=/private/ipdev/test/staging`.

Note: Do not use file or FTP URL such as (`ftp://...` or `file://...`). If the staging directory is used for a FTP server, the path specified is a directory on the FTP server. User must make sure that the path for the staging directory exists.

The properties for the FTP Receiver are in the `adapter.ini` file as `file.sender.*`.

Customizing the FTP Adapter

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

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

ReceiverCustomizer Interface

You can use the `ReceiverCustomizer` interface to customize the `TransportMessage` object that FTP adapter receives, as illustrated below:

- With the `customizeTransportMessage()` method, you can customize the `TransportMessage` object before the adapter processes it. The `TransportMessage` object represents the native message that the transport layer receives or sends. For example, there is a header in the first line of the file that you must use to dynamically modify the contents of the file that you receive.
- You can modify the message by implementing the `customizeTransportMessage()` method. The `createReplyMessage()` method is required by the HTTP adapter. For the FTP adapter, you should return a `null` value in this method.

File Structure

The following is the file structure of this interface.

```
package oracle.oai.agent.adapter.technology;
import oracle.oai.agent.adapter.transport.TransportMessage;
import oracle.oai.agent.adapter.sdk.Agent;
public interface ReceiverCustomizer {

   public void customizeTransportMessage(Agent agent,
                                          int receiverType,
                                          TransportMessage transportMessage);

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

File Summary

The following table summarizes the `ReceiverCustomizer` Interface.

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

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

Example 3–1 Example of ReceiverCustomizer

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

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

public class MyReceiverCustomizer implements ReceiverCustomizer {
```

This example describes how to remove an extra line in the incoming files.

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

Note: For debugging purposes only, the following syntax removes the first line from the payload. Details of `removeFirstLine()` is not provided.

```
      agent.logTraceMessage("payload received = " + payload, null, null, null);
      String newPayload = removeFirstLine(payload);

      try {
```

```
            transportMessage.setBody(newPayload);
         }
         catch(TransportException te) {
         . . . .
         }
      }
```

Note: For the FTP adapter, a `null` string from the following method will be returned.

```
   public String createReplyMessage(Agent agent, int status,
                                    TransportMessage receivedTransportMessage)
   {
   return null;
   }
}
```

Example 3–2 List of Methods for the TransportMessage Class

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

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

Method	Description
`public String getBodyAsString();`	Get the body of the message as String object. Return the message in String object.
`public byte[] getBodyAsBytes();`	Get the body of the message as byte array. Return the message in byte[].
`public InputStream getBodyAsInputStream();`	Get the body of the message and return an `InputStream` object representing the body of the message.

FileSenderCustomizer Interface

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

The `FileSenderCustomizer` interface extends the `SenderCustomizer` interface. You must implement the `FileSenderCustomizer` interface by implement the following two methods:

- `FileSenderCustomizer.customizeTransportMessage()`
- `FileSenderCustomizer.generateFileName()`

However, if you do not want to implement the more complicated `generateFileName()` method, you can create a class that extends the `oracle.oai.agent.adapter.technology.FileDefaultSenderCustomizer` class, which is provided in the `oai.jar` file. In this case, you only need to implement the `customizeTransportMessage()` method.

SenderCustomizer Interface

File Structure

The following is the file structure of the `SenderCustomizer` interface.

```
package oracle.oai.agent.adapter.technology;

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

public interface SenderCustomizer {
   public void customizeTransportMessage(Agent agent,
                                          TransportMessage transportMessage,
```

```
                                                MessageObject mobj,
                                                AttributeObject aobj);
}
```

File Summary

The following table summarizes the `customizeTransportMessage` method.

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

FileSenderCustomizer Interface

File Structure

The following is the file structure of the `FileSenderCustomizer` interface.

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

public interface FileSenderCustomizer extends SenderCustomizer {
   public String generateFileName (Agent agent,
                                   String rule,
                                   String app,
                                   String partitition,
                                   Date time,
                                   MessageObject mobj,
```

```
                                        AttributeObject aobj);
}
```

File Summary

The following table summarizes the `generateFileName` method.

Method	Description
`generateFileName ();`	This method generates a file name for sending a file. It contains the following parameters: `agent`—Indicates that the user can use the `Agent` object to log message. `rule`—Indicates `Rule` for generating subject. This is read from `file.sender.file_name_rule` in `adapter.ini`. `app`—Indicates the application name. `partition`—Indicates Partition. `time`—This is a `Date` object which indicates the time the OracleAS InterConnect object is received. `mobj`—Indicates a `MessageObject` passed from OracleAS InterConnect. `aobj`—Indicates an `AttributeObject` passed from OracleAS InterConnect. This method returns a string representing the file name.

FTP Error Code

This section defines the error codes that the FTP adapter returns in the event of an exception.

The error code returned by the FTP adapter corresponds to the negative acknowledgment sent by the remote FTP server to the FTP adapter. The full list of the FTP reply codes can be found in Section 4.2.2 of RFC 959.

The FTP reply code consists of three digits. There are five possible values for the first digit.

The following is the interpretation of the reply code for the negative acknowledgment.

```
4xx:   Transient Negative Completion reply
         The command is not accepted and the request action did not take place.
         The error condition is transient and can be retried.
5xx : Permanent Negative Completion reply
         The command was not accepted and the request action did not take place.
         The error condition is permanent and the user should  not retry.
```

The second digit of the reply code corresponds to different functions:

```
x0x:  syntax error
x1x: Information
x2x: Connections
x3x: Authentication and accounting
x4x: Unspecified
x5x: File system
```

See Also: *Oracle Application Server InterConnect User's Guide*

4

Frequently Asked Questions

This chapter provides answers to frequently asked questions about the FTP adapter. This chapter discusses the following topics:

- Installation Questions
- Design Time Questions

Installation Questions

The following questions address installation of the FTP adapter.

How do I know the FTP Adapter has started properly?

View the `oai.txt` file located in the appropriate timestamped subdirectory of the FTP adapter log directory:

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

If there are no exceptions, the FTP adapter has started properly.

The FTP Adapter did not start properly - what went wrong?

View the exceptions in the FTP adapter log file (`oailog.txt`). The exceptions should provide some idea about what went wrong. It is possible that the FTP adapter is unable to connect to the repository. Make sure the repository is started properly. The FTP adapter will connect to the Repository once it is started properly. You do not need to restart the Adapter.

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

Is it possible to edit the FTP adapter configuration settings created during installation?

Yes, edit the parameters in the `adapter.ini` file in the following directory:

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

> **See Also:** Chapter 2, "Installation and Configuration"

Design Time Questions

The following are design time questions for the FTP adapter.

When I change an element in iStudio, such as mappings, it seems like the FTP Adapter is using old information - what is happening?

The FTP adapter caches the information from iStudio which is stored in the repository locally for better performance in a production environment. If you change something in iStudio and want to see the change in the runtime, you need to stop the FTP adapter which are effected, delete the FTP adapter cache files and restart the FTP adapter.

The FTP adapter has a persistence directory which is located in the FTP adapter directory. Deleting this directory when the FTP adapter has been stopped should make it obtain the new metadata from the repository when started.

How do I secure any of the inifile parameters?

In order to encrypt any values specified in `.ini` file, complete the following steps:

1. Locate value to be encrypted.
2. Run the `encrypt` utility in `$OAI_HOME/bin` directory to encrypt above value.
3. Prefix the name of the parameter in the `.ini` file with `encrypted_`.
4. Replace the value with the new encrypted value from step 2.

Example: To encrypt the password for the `ftp.sender.password` parameter, replace `ftp.sender.password=ftpuser` with the following:

```
encrypted_ftp.sender.password=1124110710711065108010941084107310701071108110690
```

A

Sample Adapter.ini File

This appendix shows a sample adapter.ini file for the FTP adapter. This appendix contains this topic:

- "Sample Adapter.ini File"

> **See Also:** "FTP Adapter Configuration" on page A-2 for additional information on `adapter.ini` configuration parameters

Sample Adapter.ini File

The following code sample displays the FTP adapter `adapter.ini` file.

```
#include <../../hub/hub.ini>

// *************
// ** Adapter **
// *************

// Application (as created in iStudio) that this Adapter corresponds to.
application=myFtpApp

// Partition (as created in iStudio) that this Adapter corresponds to.
partition=

// If you want to have multiple Adapter instances for the given application with
the given part
ition, each Adapter should have an instance number.
//instance_number=2

// Bridge class
bridge_class=oracle.oai.agent.adapter.technology.TechBridge

ota.type=D3L


// define the ftp sending endpoint
// For ftp, ota.send.endpoint=ftp://<host name>/<path name>
// For file, ota.send.endpoint=file://<host name>/<path name>
//
ota.send.endpoint= ftp://foo.s.com/private/ipdev1/test/d3l/inbound

// define the ftp receiving endpoint
// For ftp, ota.send.endpoint=ftp://<host name>/<path name>
// For file, ota.send.endpoint=file://<host name>/<path name>
//
ota.receive.endpoint=ftp://foo.s.com/private/ipdev1/test/d3l/inbound


//--------------------------------------
// ftp Sender initialization variables
//--------------------------------------

// ftp user (mandatory if ftp is used)
```

```
// file.sender.user=ipdev1
file.sender.user=ipdev1


// ftp user password (mandatory if ftp is used)
//file.sender.password=ipdev1
file.sender.password=ipwelcome


// file type (ASCII or BINARY)
//file.sender.type=BINARY
file.sender.type=ASCII

// proxy host
//file.sender.proxy_host=

// proxy port
//file.sender.proxy_port=
//staging directory
//file.sender.staging_directory =/tmp

//sender customizer class
//file.sender.customizer_class = MySenderCustomizer

//--------------------------------------
// ftp receiver initialization variables
//--------------------------------------

// ftp user (mandatory if ftp is used)
//file.receiver.user=ipdev1
file.receiver.user=ipdev1

// ftp user password (mandatory if ftp is used)
//file.receiver.password=ipdev1
file.receiver.password=ipwelcome

// file type (ASCII or BINARY)
//file.receiver.type=BINARY
file.receiver.type=BINARY

// proxy host
//file.receiver.proxy_host=

// proxy port
//file.receiver.proxy_port=
```

```
//receiver customizer class
//file.receiver.customizer_class = MyReceiverCustomizer

// define where to put the
// file that cannot be processed
// properly.
//file.receiver.exception_dir=


// define how often to poll
// the message source (in milli seconds)
file.receiver.polling_interval=60000

// define maximum number of messages
// retrieved in each polling session
file.receiver.max_msgs_retrieved=30

// D3L initialization variables
ota.d3ls=person2.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=myFTPApp
agent_message_selector=recipient_list like '%,myFTPApp,%'
// Only provide values for the next two parameters if you have multiple Adapter
instances for t
he given application with the given partition.
//agent_reply_subscriber_name=
//agent_reply_message_selector=

// Set this to false if you want to turn off all tracking of messages (if true,
messages which
have tracking fields set in iStudio will be tracked)
agent_tracking_enabled=true

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

```
// By default, Adapters use an OAI specific DTD for all messages sent to the Hub
since other OA
I Adapters will be picking up the messages from the Hub and know how to
interpret them. This sh
ould be set to true if for every message, you would like to use the DTD imported
for that messa
ge'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 (cach
e 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 table
s 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), de
mand (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 de
mand) 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 star
ted, 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


  //////////////////
 // End Comments //
//////////////////
```

Index

A

B

C

D

E

F

H

I

J

L

M

N

O

P

R

S

X