

9.0

Managed File Transfer

IBM

Note

Before using this information and the product it supports, read the information in [“Notices” on page 917.](#)

This edition applies to version 9 release 0 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2007, 2024.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

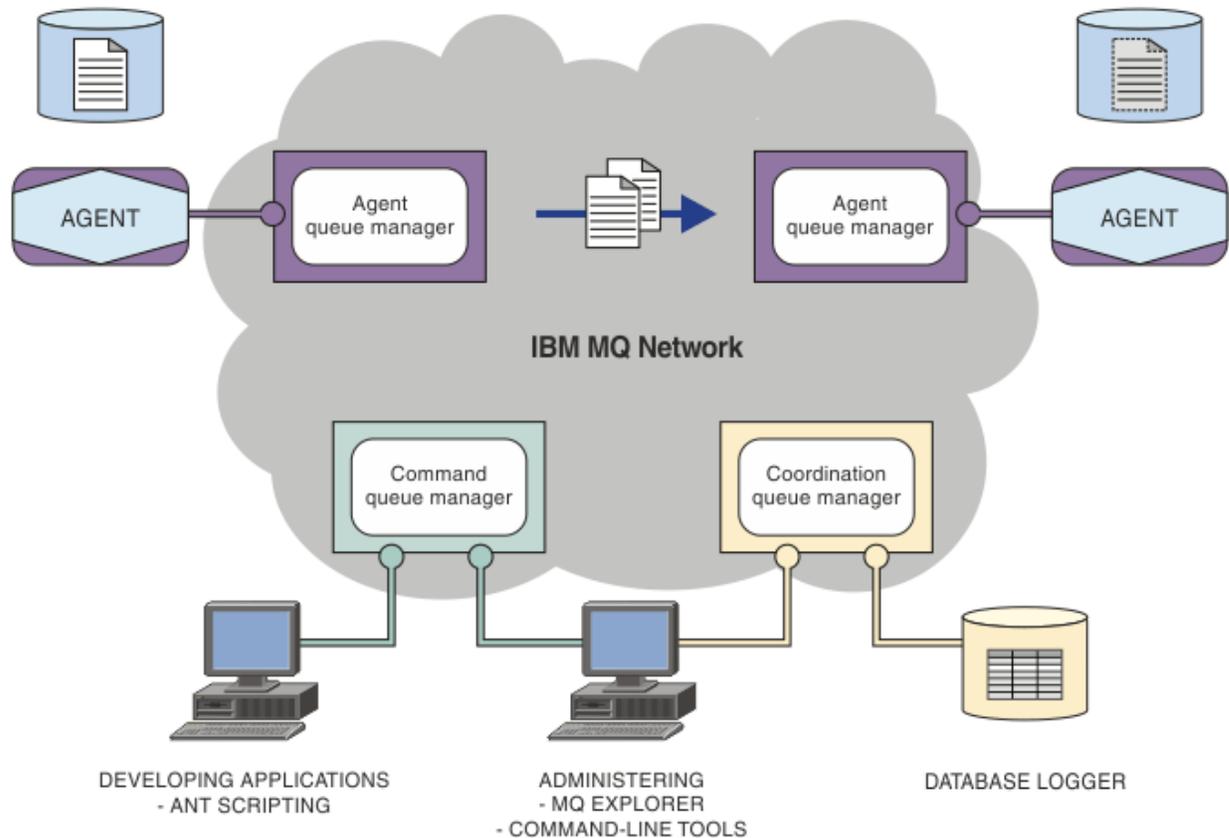
Managed File Transfer.....	5
MFT overview.....	7
MFT introduction.....	7
How does MFT work with IBM MQ?.....	9
MFT topology overview.....	9
MFT REST API overview.....	10
What's new and changed in MFT in IBM MQ 9.0?.....	11
Migration considerations for MFT.....	12
MFT scenario overview.....	27
MFT common topologies.....	27
Configuring the base server.....	31
Securing MFT.....	38
MFT and IBM MQ connection authentication.....	39
MFT sandboxes.....	45
Configuring SSL or TLS encryption for MFT.....	51
Connecting to a queue manager in client mode with channel authentication.....	52
Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node.....	53
Configuring MFT.....	55
MFT configuration options on Multiplatforms.....	56
MFT configuration options on z/OS.....	58
Configuring the Redistributable Managed File Transfer Agent.....	58
Creating an MFT Agent or Logger command data set.....	62
Configuring Managed File Transfer for z/OS.....	63
Configuring MFT on IBM i.....	89
Configuring MFT for first use.....	90
Configuring an MFT logger.....	100
Configuring the Connect:Direct bridge.....	135
Configuring MFT agents with MSCS.....	145
Administering MFT.....	146
Starting an MFT agent.....	146
Starting a new file transfer.....	152
Creating a scheduled file transfer.....	155
Working with pending file transfers	156
Triggering a file transfer.....	156
Monitoring file transfers that are in progress.....	158
Viewing the status of file transfers in the Transfer Log.....	160
Monitoring MFT resources.....	162
Working with file transfer templates.....	186
Transferring data from files to messages.....	189
Transferring data from messages to files.....	197
Listing MFT agents.....	203
Stopping an MFT agent.....	203
The protocol bridge.....	204
The Connect:Direct bridge.....	220
Working with MFT from IBM Integration Bus.....	235
MFT recovery and restart.....	235
Timeout option for file transfers in recovery.....	236
Developing applications for MFT.....	239
Specifying programs to run with MFT.....	239
Using Apache Ant with MFT.....	241
Customizing MFT with user exits.....	245
Controlling MFT by putting messages on the agent command queue.....	258

Troubleshooting MFT.....	259
MFT general troubleshooting.....	259
Troubleshooting the Connect:Direct bridge.....	328
MFT reference.....	334
MFT security reference.....	334
Which MFT commands and processes connect to which queue manager.....	349
MFT commands.....	356
MFT configuration reference.....	511
MFT administration reference.....	599
MFT developing applications reference.....	818
MFT diagnostic messages.....	915
Notices.....	917
Programming interface information.....	918
Trademarks.....	918

Managed File Transfer

Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.

You can use Managed File Transfer to build a customized, scalable, and automated solution that enables you to manage, trust, and secure file transfers. Managed File Transfer eliminates costly redundancies, lowers maintenance costs, and maximizes your existing IT investments.



The diagram shows a simple Managed File Transfer topology. There are two agents, each connect to their own agent queue manager in an IBM MQ network. A file is transferred from the agent on the one side of the diagram, through the IBM MQ network, to the agent on the other side of the diagram. Also in the IBM MQ network are the coordination queue manager and a command queue manager. Applications and tools connect to these queue managers to configure, administer, operate, and log Managed File Transfer activity in the IBM MQ network.

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service, or Managed File Transfer Tools. For more information, see [Managed File Transfer product options](#).

You can use Managed File Transfer to perform the following tasks:

- Create managed file transfers
 - Create new file transfers from IBM MQ Explorer on Linux® or Windows platforms.
 - Create new file transfers from the command line on all supported platforms.
 - Integrate file transfer function into the Apache Ant tool.

- Write applications that control Managed File Transfer by putting messages on agent command queues.
- Schedule file transfers to take place at a later time. You can also trigger scheduled file transfers based on a range of file system events, for example a new file being created.
- Continually monitor a resource, for example a directory, and when the contents of that resource meet some predefined condition, start a task. This task can be a file transfer, an Ant script, or a JCL job.
- Transfer files to and from IBM MQ queues.
- Transfer files to and from FTP, FTPS, or SFTP servers.
- Transfer files to and from Connect:Direct® nodes.
- Transfer both text and binary files. Text files are automatically converted between the code pages and end-of-line conventions of the source and destination systems.
- Transfers can be secured, using the industry standards for Secure Socket Layer (SSL) based connections.
- View transfers in progress and log information about all transfers in your network
 - View the status of transfers in progress from IBM MQ Explorer on Linux or Windows platforms.
 - Check the status of completed transfers by using the IBM MQ Explorer on Linux or Windows platforms.
 - Use the Managed File Transfer database logger feature to save log messages to a Db2® or Oracle database.

Managed File Transfer is built on IBM MQ, which provides assured, once-only delivery of messages between applications. You can take advantage of various features of IBM MQ. For example, you can use channel compression to compress the data that you send between agents over IBM MQ channels and use SSL channels to secure the data that you send between agents. Files are transferred reliably and can tolerate the failure of the infrastructure over which the file transfer is carried out. If you experience a network outage, the file transfer restarts from where it left off when connectivity is restored.

By consolidating file transfer with your existing IBM MQ network, you can avoid spending the resources required to maintain two separate infrastructures. If you are not already an IBM MQ customer, by creating an IBM MQ network to support Managed File Transfer you are building the backbone for a future SOA implementation. If you are already an IBM MQ customer, Managed File Transfer can take advantage of your existing IBM MQ infrastructure including IBM MQ Internet Pass-Thru and IBM Integration Bus.

Managed File Transfer integrates with a number of other IBM products:

IBM Integration Bus

Process files that have been transferred by Managed File Transfer as part of an IBM Integration Bus flow. For more information, see [“Working with MFT from IBM Integration Bus”](#) on page 235.

IBM Sterling Connect:Direct

Transfer files to and from an existing Connect:Direct network by using the Managed File Transfer Connect:Direct bridge. For more information, see [“The Connect:Direct bridge”](#) on page 220.

IBM Tivoli® Composite Application Manager

IBM Tivoli Composite Application Manager provides an agent that you can use to monitor information that is published to the coordination queue manager.

Related concepts

[Managed File Transfer product options](#)

[“MFT topology overview”](#) on page 9

An overview of how Managed File Transfer agents are connected with the coordination queue manager in an IBM MQ network.

[“How does MFT work with IBM MQ?”](#) on page 9

Managed File Transfer interacts in a number of ways with IBM MQ.

MFT overview

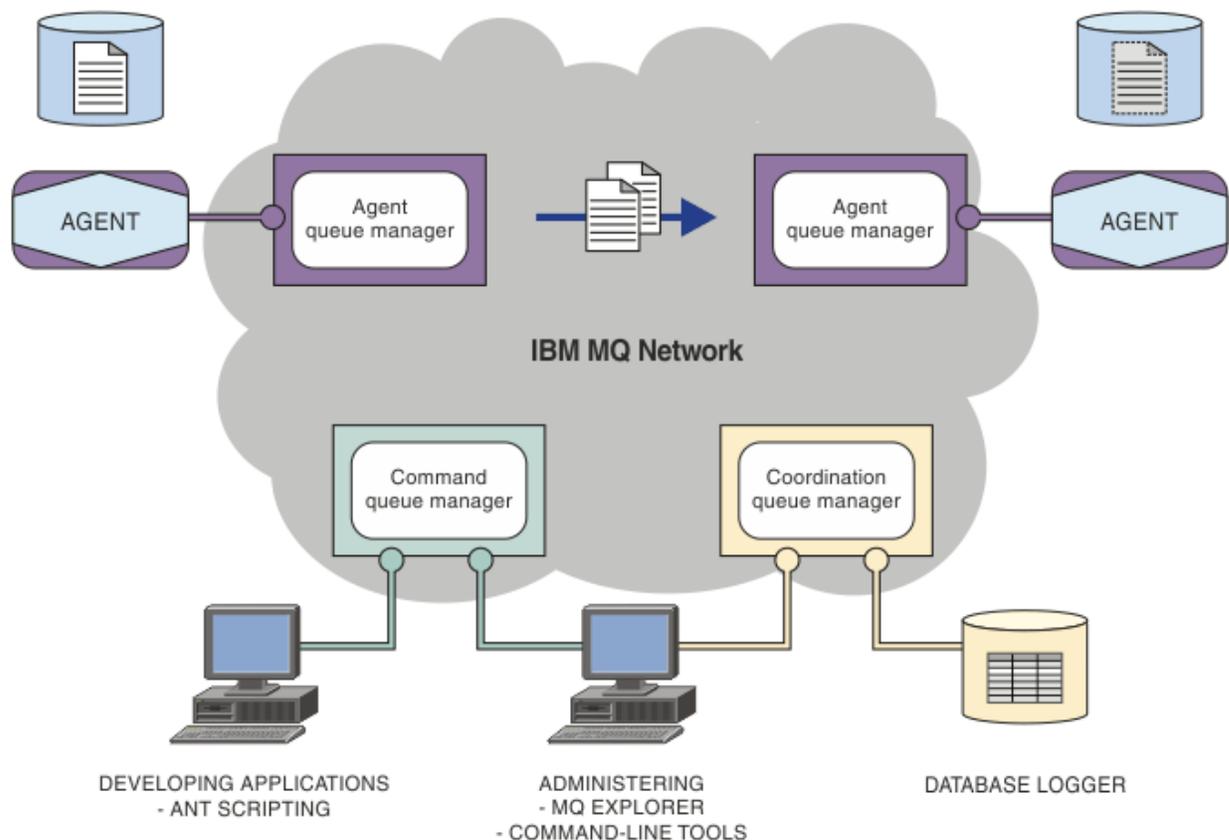
This section provides introductory information that you can use to get started with Managed File Transfer.

- [“Managed File Transfer” on page 5](#)
- [Managed File Transfer product options](#)
- [“MFT topology overview” on page 9](#)
- [“What's new and changed in MFT in IBM MQ 9.0?” on page 11](#)

Managed File Transfer

Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.

You can use Managed File Transfer to build a customized, scalable, and automated solution that enables you to manage, trust, and secure file transfers. Managed File Transfer eliminates costly redundancies, lowers maintenance costs, and maximizes your existing IT investments.



The diagram shows a simple Managed File Transfer topology. There are two agents, each connect to their own agent queue manager in an IBM MQ network. A file is transferred from the agent on the one side of the diagram, through the IBM MQ network, to the agent on the other side of the diagram. Also in the IBM MQ network are the coordination queue manager and a command queue manager. Applications and tools connect to these queue managers to configure, administer, operate, and log Managed File Transfer activity in the IBM MQ network.

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are Managed File Transfer Agent, Managed File Transfer Logger, Managed

File Transfer Service, or Managed File Transfer Tools. For more information, see [Managed File Transfer product options](#).

You can use Managed File Transfer to perform the following tasks:

- Create managed file transfers
 - Create new file transfers from IBM MQ Explorer on Linux or Windows platforms.
 - Create new file transfers from the command line on all supported platforms.
 - Integrate file transfer function into the Apache Ant tool.
 - Write applications that control Managed File Transfer by putting messages on agent command queues.
 - Schedule file transfers to take place at a later time. You can also trigger scheduled file transfers based on a range of file system events, for example a new file being created.
 - Continually monitor a resource, for example a directory, and when the contents of that resource meet some predefined condition, start a task. This task can be a file transfer, an Ant script, or a JCL job.
 - Transfer files to and from IBM MQ queues.
 - Transfer files to and from FTP, FTPS, or SFTP servers.
 - Transfer files to and from Connect:Direct nodes.
 - Transfer both text and binary files. Text files are automatically converted between the code pages and end-of-line conventions of the source and destination systems.
 - Transfers can be secured, using the industry standards for Secure Socket Layer (SSL) based connections.
- View transfers in progress and log information about all transfers in your network
 - View the status of transfers in progress from IBM MQ Explorer on Linux or Windows platforms.
 - Check the status of completed transfers by using the IBM MQ Explorer on Linux or Windows platforms.
 - Use the Managed File Transfer database logger feature to save log messages to a Db2 or Oracle database.

Managed File Transfer is built on IBM MQ, which provides assured, once-only delivery of messages between applications. You can take advantage of various features of IBM MQ. For example, you can use channel compression to compress the data that you send between agents over IBM MQ channels and use SSL channels to secure the data that you send between agents. Files are transferred reliably and can tolerate the failure of the infrastructure over which the file transfer is carried out. If you experience a network outage, the file transfer restarts from where it left off when connectivity is restored.

By consolidating file transfer with your existing IBM MQ network, you can avoid spending the resources required to maintain two separate infrastructures. If you are not already an IBM MQ customer, by creating an IBM MQ network to support Managed File Transfer you are building the backbone for a future SOA implementation. If you are already an IBM MQ customer, Managed File Transfer can take advantage of your existing IBM MQ infrastructure including IBM MQ Internet Pass-Thru and IBM Integration Bus.

Managed File Transfer integrates with a number of other IBM products:

IBM Integration Bus

Process files that have been transferred by Managed File Transfer as part of an IBM Integration Bus flow. For more information, see [“Working with MFT from IBM Integration Bus” on page 235](#).

IBM Sterling Connect:Direct

Transfer files to and from an existing Connect:Direct network by using the Managed File Transfer Connect:Direct bridge. For more information, see [“The Connect:Direct bridge” on page 220](#).

IBM Tivoli Composite Application Manager

IBM Tivoli Composite Application Manager provides an agent that you can use to monitor information that is published to the coordination queue manager.

Related concepts

[Managed File Transfer product options](#)

[“MFT topology overview” on page 9](#)

An overview of how Managed File Transfer agents are connected with the coordination queue manager in an IBM MQ network.

[“How does MFT work with IBM MQ?” on page 9](#)

Managed File Transfer interacts in a number of ways with IBM MQ.

How does MFT work with IBM MQ?

Managed File Transfer interacts in a number of ways with IBM MQ.

- Managed File Transfer transfers files between agent processes by dividing each file into one or more messages and transmitting the messages through your IBM MQ network.
- The agent processes move file data by using nonpersistent messages to minimize the impact on your IBM MQ logs. By communicating with one another the agent processes regulate the flow of messages containing file data. This prevents messages containing file data building up on IBM MQ transmission queues and ensures that if any of the nonpersistent messages are not delivered, the file data is sent again.
- Managed File Transfer agents use a number of IBM MQ queues. For more information, see [“MFT system queues and the system topic” on page 596](#).
- Although some of these queues are strictly for internal use, an agent can accept requests in the form of specially formatted command messages sent to a specific queue that the agent reads from. Both the command-line commands and the IBM MQ Explorer plugin send IBM MQ messages to the agent to instruct the agent to perform the wanted action. You can write IBM MQ applications that interact with the agent in this way. For more information, see [“Controlling MFT by putting messages on the agent command queue” on page 258](#).
- Managed File Transfer agents send information about their state and the progress and outcome of transfers to an MQ queue manager that has been designated as the coordination queue manager. This information is published by the coordination queue manager and can be subscribed to by applications that want to monitor transfer progress or keep records of the transfers that have occurred. Both the command-line commands and the IBM MQ Explorer plug-in can use the information that is published. You can write IBM MQ applications that use this information. For more information about the topic that the information is published to, see [“SYSTEM.FTE topic” on page 594](#).
- Key components of Managed File Transfer take advantage of the capability of IBM MQ queue managers to store and forward messages. This means that if you suffer an outage, unaffected parts of your infrastructure can continue to transfer files. This extends to the coordination queue manager, where a combination of store and forward and durable subscriptions allow the coordination queue manager to tolerate becoming unavailable without losing key information about the file transfers that have taken place.

MFT topology overview

An overview of how Managed File Transfer agents are connected with the coordination queue manager in an IBM MQ network.

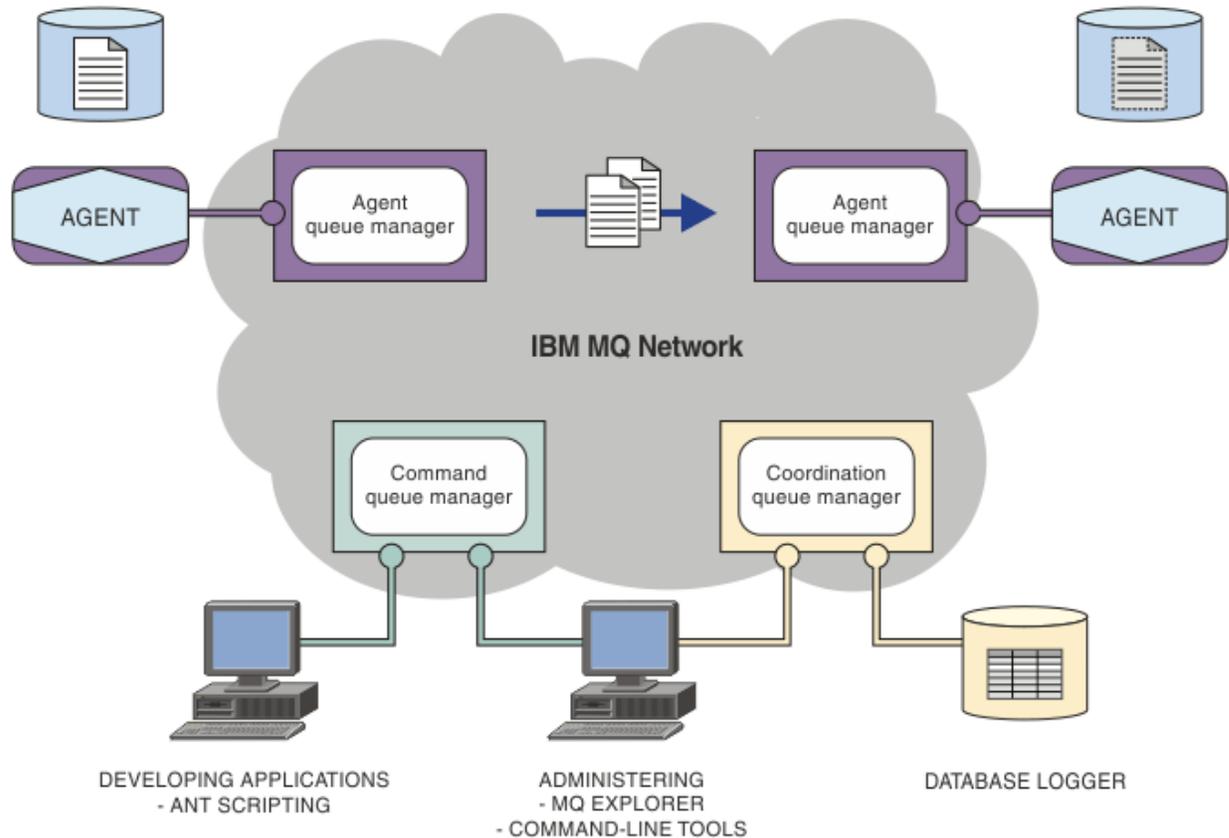
Managed File Transfer agents send and receive the files that are transferred. Each agent has its own set of queues on its associated queue manager and the agent is attached to its queue manager in either bindings or client mode. An agent can also use the coordination queue manager as its queue manager.

The coordination queue manager broadcasts audit and file transfer information. The coordination queue manager represents a single point for the collection of agent, transfer status, and transfer audit information. The coordination queue manager is not required to be available in order for transfers to take place. If the coordination queue manager temporarily becomes unavailable, transfers continue as normal. Audit and status messages are stored in the agent queue managers until the coordination queue manager became available, and can then be processed as normal.

Agents register with the coordination queue manager and publish their details to that queue manager. This agent information is used by the Managed File Transfer plugin to enable the start of transfers from the IBM MQ Explorer. The agent information collected on the coordination queue manager is also used by the commands to display agent information and agent status.

Transfer status and transfer audit information is published on the coordination queue manager. The transfer status and transfer audit information is used by the Managed File Transfer plug-in to monitor the progress of transfers from the IBM MQ Explorer. The transfer audit information stored on the coordination queue manager can be retained to provide auditability.

The command queue manager is used to connect to the IBM MQ network and is the queue manager connected to when you issue Managed File Transfer commands.



Related concepts

[“Managed File Transfer” on page 5](#)

Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.

[“How does MFT work with IBM MQ?” on page 9](#)

Managed File Transfer interacts in a number of ways with IBM MQ.

[“MFT scenario overview” on page 27](#)

This section lists common Managed File Transfer topologies together with a scenario that sets up the system and transfers a test message.

V 9.0.5 MFT REST API overview

An overview of the REST API enhancements for Managed File Transfer.

From IBM MQ 9.0.5, the REST API adds support for certain Managed File Transfer commands, including listing transfers, and details about file transfer agents.

See [further enhancements to the REST API](#) for more information.

What's new and changed in MFT in IBM MQ 9.0?

Links to information about the main new and changed Managed File Transfer functions in IBM MQ 9.0.

IBM MQ 9.0.0

► V 9.0.0

- ► V 9.0.0 [Managed File Transfer Protocol Bridge Agent logging enhancements](#)
- ► V 9.0.0 [Removal of Managed File Transfer Web Gateway support](#)

IBM MQ 9.0.x Continuous Delivery releases

CD

- ► Windows ► Linux ► V 9.0.1 [Managed File Transfer configurable recovery timeout for stalled transfers](#)
- ► Windows ► Linux ► V 9.0.1 [Managed File Transfer Agent redistributable package](#)
- ► Windows ► Linux ► V 9.0.1 [Changes to **DISPLAY CHSTATUS** output values for remote Managed File Transfer Agents](#)
- ► Windows ► Linux ► V 9.0.1 [Changes to message to file transfers in Managed File Transfer](#)
- ► V 9.0.1 [New MFT agent property `additionalWildcardSandboxChecking`](#)
- ► Windows ► Linux ► V 9.0.2 [Improved naming for subscriptions created by the IBM MQ Explorer MFT plugin](#)
- ► Windows ► Linux ► V 9.0.2 [Improved information about the MFT agent when the last reported status is UNKNOWN](#)
- ► V 9.0.2 ► z/OS ► MQ Adv. VUE [New agent property `adminGroup` for use with MFT agents on z/OS®](#)
- ► Windows ► V 9.0.3 ► Linux ► z/OS [Managed File Transfer resource monitor event logging](#)
- [Changes in z/OS product identifier \(PID\) recording](#)
- ► V 9.0.3 ► z/OS ► MQ Adv. VUE [Agent remote connections with IBM MQ Advanced for z/OS, Value Unit Edition](#)
- ► Windows ► Linux ► z/OS [Change to behavior of Managed File Transfer `fteCleanAgent` command](#)
- ► Windows ► Linux [Corrected behavior for message `PublicationFormat=mixed` in the Managed File Transfer installation properties](#)
- ► Windows ► V 9.0.3 ► Linux ► z/OS [Enabling MQCSP authentication for Managed File Transfer](#)
- ► Windows ► Linux ► V 9.0.4 ► AIX [Ability to run the Managed File Transfer Logger in client mode](#)
- ► Multi ► V 9.0.5 [“MFT REST API overview” on page 10](#)

IBM MQ 9.0.0.x Long Term Support

LTS

- [V 9.0.0.1](#) **fteMigrateAgent** command changes
- [V 9.0.0.1](#) New MFT agent property `addCommandPathToSandbox`
- [V 9.0.0.1](#) New MFT agent property `additionalWildcardSandboxChecking`
- [V 9.0.0.1](#) [z/OS](#) New agent property `adminGroup` for use with MFT agents on z/OS
- [V 9.0.0.2](#) Change to behavior of Managed File Transfer **fteCleanAgent** command
- [V 9.0.0.2](#) Enabling MQCSP authentication in Managed File Transfer
- [V 9.0.0.2](#) New **messagePublicationFormat** property for Managed File Transfer `installation.properties` file
- [V 9.0.0.2](#) Comparison check of transfer identifier and value of `groupId` attribute restored for message-to-file transfers
- [V 9.0.5](#) Ability to use colon separated lists in `fteSetAgentTraceLevel` and `fteSetLoggerTracelevel` commands

Related reference

- [V 9.0.0](#) What's new and changed in IBM MQ 9.0.0
- [CD](#) What's new and changed in IBM MQ 9.0.x Continuous Delivery
- [LTS](#) What's changed in IBM MQ 9.0.x Long Term Support

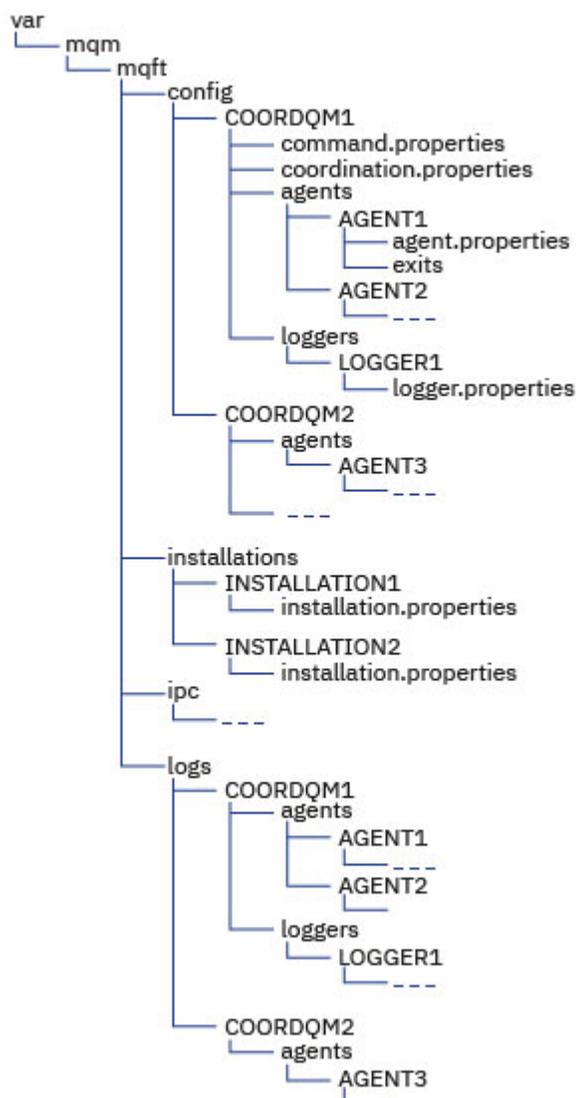
Migration considerations for MFT

If you are planning to migrate Managed File Transfer, review the following information.

Configuration layout directly after installation in MFT

The diagram shows the configuration layout directly after installation as it is in IBM MQ.

IBM MQ Managed File Transfer configuration



Default configuration directories:

Information is split over four separate sub-directories: config, installations, ipc, and logs.

Default product root directories (MQ_DATA_PATH) are as follows:

- UNIX systems: /var/mqm
- Linux systems: /var/mqm
- Windows: the location of the configuration directory depends on the location of your primary IBM MQ installation. The default locations for primary installations are as follows:
 - 32-bit: C:\Program Files (x86)\IBM\WebSphere MQ
 - 64-bit: C:\Program Files\IBM\MQ

Configuration sub-directories are as follows:

- The `MQ_DATA_PATH/mqft/config` directory contains the parts of the configuration that are read-only for Managed File Transfer processes. For example, `agent.properties` and `command.properties`.

- The `MQ_DATA_PATH/mqft/installations` directory contains configuration information for each installation. The content of this directory is equivalent to the content of the `wmqfte.properties` file.
- The `MQ_DATA_PATH/mqft/ipc` directory contains IPC resources used internally to communicate between the Managed File Transfer components. Applicable to UNIX and Linux systems only.
- The `MQ_DATA_PATH/mqft/logs` directory contains the parts of the configuration that are written by Managed File Transfer processes. For example, trace information and log files.

installation.properties file

Linux **UNIX** On UNIX and Linux systems, the default location is `MQ_DATA_PATH/mqft/installations/installation_name`

Windows On Windows, the default location is `MQ_DATA_PATH\mqft\installations\installation_name`

logger.properties file

This file incorporates property information for stand-alone file loggers, stand-alone database loggers, and Java EE database loggers.

The default location is `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name`.

Security changes

For IBM WebSphere® MQ 7.5, or later, only users who are administrators (members of the `mqm` group) can run the following list of **fte** commands:

- [“fteChangeDefaultConfigurationOptions” on page 365](#)
- [“fteCreateAgent \(create an MFT agent\)” on page 371](#)
- [“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)” on page 375](#)
- [“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 384](#)
- [“fteCreateLogger \(create an MFT file or database logger\)” on page 389](#)
- [“fteDeleteAgent” on page 442](#)
- [“fteDeleteLogger” on page 444](#)
- [“fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later” on page 462](#)
- [“fteMigrateConfigurationOptions: migrate an FTE 7.0 configuration to MQ 7.5 or later” on page 464](#)
- [“fteMigrateLogger: migrate an FTE 7.0 database logger to MQ 7.5 or later” on page 466](#)
- [“fteModifyAgent \(run an MFT agent as a Windows service\)” on page 469](#)
- [“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#)
- [“fteSetupCommands: create the MFT command.properties file” on page 490](#)
- [“fteSetupCoordination” on page 492](#)

Multi When using IBM WebSphere MQ 7.5 or later on [Multiplatforms](#), only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

V 9.0.2 **z/OS** For z/OS, for Continuous Delivery from IBM MQ 9.0.2, the **fteSetAgentTraceLevel** command can be run by either:

- The same userid that the agent process is running as.
- Members of the group specified by the agent property **adminGroup**.

V 9.0.0.1 This also applies to the Long Term Support release from IBM MQ 9.0.0 Fix Pack 1.

V 9.0.2**z/OS**

For z/OS, for Continuous Delivery from IBM MQ 9.0.2, if the **fteShowAgentDetails** command is run to display information about a local agent, then the command can be run by either:

- The same userid that the agent process is running as.
- Members of the group specified by the agent property **adminGroup**.

V 9.0.0.1

This also applies to the Long Term Support release from IBM MQ 9.0.0 Fix Pack 1.

For more information, see the **adminGroup** property in [“The MFT agent.properties file” on page 526](#).

Security changes in IBM MQ 8.0

If you are running Managed File Transfer on IBM WebSphere MQ 7.0, and migrate to IBM MQ 8.0, the user Id information in the `MQMFTCredentials.xml` file is passed to the queue manager, but will not be acted upon.

This is because the passing of user Id and password information only is supported in IBM MQ 8.0.

commandPath and agent sandboxes

For IBM MQ 8.0 and later, if an agent has been configured with an agent sandbox and the agent property `commandPath` has been set, then the directories specified by `commandPath` are automatically added to the denied paths when the agent starts. If the `commandPath` property is set on an agent which is not configured with an agent sandbox, then a new sandbox is set up automatically and the directories specified by the `commandPath` are added to the denied directories when the agent starts.

If the `commandPath` property is set on an agent which is not configured with an agent sandbox, then a new sandbox is set up automatically and the directories specified by the `commandPath` are added to the denied directories when the agent starts.

For more information about the `commandPath` property, see [“commandPath MFT property” on page 347](#) and [“The MFT agent.properties file” on page 526](#).

commandPath and user sandboxes

For IBM MQ 8.0 and later, if an agent has been configured with one or more user sandboxes, and has the agent property `commandPath` set, then the directories specified by `commandPath` (and all of their subdirectories) are automatically added as `<exclude>` elements to the `<read>` and `<write>` elements for each user sandbox when the agent starts up.

For more information about the `commandPath` property, see [“commandPath MFT property” on page 347](#) and [“The MFT agent.properties file” on page 526](#).

Migrating MFT agents that run as a Windows service from IBM WebSphere MQ 7.5 to IBM MQ 9.0

Between IBM WebSphere MQ 7.5, IBM MQ 8.0, and IBM MQ 9.0, the default IBM MQ installation path has changed on the Windows platform.

If a queue manager is being migrated from IBM WebSphere MQ 7.5 to IBM MQ 8.0 or IBM MQ 9.0, any applications that are running on the same system as the queue manager must be reconfigured to load the IBM MQ libraries from the new installation location. This includes any IBM MQ Managed File Transfer agents that are running as a Windows service.

Carry out the following procedure:

1. Stop the agents that are associated with the IBM WebSphere MQ 7.5 queue manager. See [stopping an MFT agent](#) for more information. For example issue the following command:

```
fteStopAgent <agent_name>
```

2. Modify the agent to remove the service definition of the agent. See [run an MFT agent as a Windows service](#) for more information.

For example, issue the following command:

```
fteModifyAgent -agentName <agent_name>
```

3. Next, migrate the agent queue manager from IBM WebSphere MQ 7.5 to the later version by using the **setmqm** command. See [associating a queue manager with an installation](#) for more information about the **setmqm** command.
4. Finally, modify the agent to reconfigure the agent to run as a Windows service again, by using the **fteModifyAgent** command.

For example:

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

Migrating an IBM WebSphere MQ File Transfer Edition 7.0 agent to IBM WebSphere MQ 7.5 or later

Use the **fteMigrateAgent** command to migrate an IBM WebSphere MQ File Transfer Edition 7.0 agent to IBM WebSphere MQ 7.5, or later. If an agent is configured to run as a Windows service, you must complete the optional steps in this task.

Before you begin

V 9.0.0 From IBM MQ 9.0, Managed File Transfer does not support web agents. If you attempt to use the **fteMigrateAgent** command to migrate a web agent from an earlier release to IBM MQ 9.0, an error message is displayed to explain that the migration of a web agent is not supported.

Identify instances of the IBM WebSphere MQ File Transfer Edition agent installed as part of IBM WebSphere MQ File Transfer Edition.

IBM i The information in this topic does not apply to IBM i. If you are using IBM i, migrate your queue manager to IBM MQ 8.0 before using Managed File Transfer in IBM MQ 8.0.

About this task

To migrate to IBM WebSphere MQ 7.5 or later, first stop the agent, then migrate the queue manager installation to IBM WebSphere MQ 7.5 or later, and additionally select the File Transfer components. When the queue manager has been migrated, you can migrate the agent's configuration to the IBM WebSphere MQ 7.5 or later, installation using the **fteMigrateAgent** command. Start the migrated agent using the IBM WebSphere MQ 7.5 or later, queue manager.

If the agent is connecting to its queue manager using IBM WebSphere MQ bindings mode and the queue manager is at IBM WebSphere MQ 7.0.1 Fix Pack 6 or later, you can alternatively perform a side-by-side migration by using multiple installations and the steps listed below. If you do not wish to perform a side-by-side migration, instead migrate the queue manager and agent using the steps as described above.

If the agent is connecting to its queue manager using IBM WebSphere MQ bindings mode and the queue manager is at IBM WebSphere MQ 7.0.1 Fix Pack 5 or earlier, you must either migrate this installation to IBM WebSphere MQ 7.0.1 Fix Pack 6, to permit a side-by-side migration, or you must migrate this installation directly to IBM WebSphere MQ 7.5 or later.

If the agent is connecting to its queue manager across a network as an IBM WebSphere MQ client, you can migrate the agent to IBM WebSphere MQ 7.5 or later by completing step “7” on page 17 only.

If the agent is configured as a Windows service you must run the command with the **-f** parameter. For more information, see [“fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later”](#) on page 462.

Procedure

1. Install IBM WebSphere MQ 7.5 or later alongside the existing queue manager installation, selecting the MQ Server and File Transfer components.
2. Stop the IBM WebSphere MQ File Transfer Edition 7.0.x agent.
3. Move the queue manager to the IBM WebSphere MQ 7.5, or later installation.
4. Use the **fteMigrateAgent** command to migrate the agent configuration from the IBM WebSphere MQ File Transfer Edition 7.0.x installation to the agent capability integrated with IBM WebSphere MQ 7.5 or later.
5. Optional: If the IBM WebSphere MQ File Transfer Edition 7.0.x agent is configured as a Windows service, complete the following optional steps:
 - a) Use the **fteModifyAgent** command to reconfigure the agent so that it is no longer a Windows service.
 - b) Use the **fteModifyAgent** command from the IBM WebSphere MQ 7.5 or later, installation to configure the IBM WebSphere MQ 7.5 or later, version of the agent to be a Windows service.
6. Start the IBM WebSphere MQ 7.5 or later, agent.
7. Optional: To migrate an agent connecting to its queue manager across a network as a client:
 - a) Install IBM WebSphere MQ 7.5 or later, onto the system
 - b) Use the **fteMigrateAgent** command to migrate the agent configuration from the WMQFTE 7.0.x installation to the agent capability integrated with IBM WebSphere MQ 7.5 or later.
 - c) Stop the IBM WebSphere MQ File Transfer Edition 7.0.x agent.
 - d) If the IBM WebSphere MQ File Transfer Edition 7.0.x agent is configured as a Windows service, use the **fteModifyAgent** command from the IBM WebSphere MQ File Transfer Edition 7.0.x installation to re-configure the agent so that it is not run as a Windows service
 - e) If the IBM WebSphere MQ File Transfer Edition 7.0.x agent is configured as a Windows service, use the **fteModifyAgent** command from the IBM WebSphere MQ 7.5 or later installation to configure the IBM WebSphere MQ 7.5 or later agent as a Windows service.
 - f) Start the IBM WebSphere MQ 7.5 or later agent.

Migrating an FTE 7.0 database logger to MQ 7.5 or later

Use the **fteMigrateLogger** command to migrate a stand-alone IBM WebSphere MQ File Transfer Edition 7.0 database logger to IBM WebSphere MQ 7.5, or later. If you have configured the database logger as a Windows service, you must complete extra migration steps.

Before you begin

Identify instances of the stand-alone database logger. You must update database logger instances before other parts of the Managed File Transfer network so that these instances can correctly process the latest versions of the transfer log messages they receive.

About this task

To migrate directly to IBM WebSphere MQ 7.5, or later, first stop the logger and then migrate this installation to the required version. When this migration is complete, use the **fteMigrateLogger** command to migrate the database logger configuration to required version of IBM MQ.

If the queue manager is IBM WebSphere MQ File Transfer Edition 7.0.1 Fix Pack 6 or later, you can alternatively perform a side-by-side migration by using multiple installations as detailed in the following steps. If you do not wish to perform a side-by-side migration, instead migrate the queue manager and logger using the steps as described above.

If the database logger is connected to an IBM WebSphere MQ File Transfer Edition 7.0.1 Fix Pack 5 or earlier queue manager, you must first either migrate this installation to IBM WebSphere MQ File Transfer

Edition 7.0.1 Fix Pack 6, to permit a side-by-side migration, or you must migrate this installation directly to IBM WebSphere MQ 7.5, or later, and additionally select the File Transfer components.

If the database logger is configured as a Windows service you must run the **fteMigrateLogger** command with the **-f** parameter. For more information, see [“fteMigrateLogger: migrate an FTE 7.0 database logger to MQ 7.5 or later”](#) on page 466.

Procedure

1. Install IBM WebSphere MQ 7.5, or later, alongside the existing IBM WebSphere MQ File Transfer Edition 7.0 installation, selecting the MQ Server and Managed File Transfer Service components.
2. Stop the IBM WebSphere MQ File Transfer Edition 7.0 database logger.
3. Move the queue manager to the IBM WebSphere MQ 7.5, or later installation.
4. Use the **fteMigrateLogger** command to migrate the database logger configuration from the IBM WebSphere MQ File Transfer Edition 7.0 installation to the IBM WebSphere MQ 7.5, or later database logger.
5. Create the tables needed for the database logger to start by running the sql scripts in *MQ_INSTALLATION_PATH/mqft/sql*. There is one script for each increase in product level, run all that are appropriate in order.

You must run the upgrade sql scripts in version order, starting with their current level of IBM WebSphere MQ File Transfer Edition or Managed File Transfer. The available scripts, where *** can be Db2, Oracle, or z/OS, are as follows:

- `ftelog_tables_***_701-702.sql`
- `ftelog_tables_***_702-703.sql`
- `ftelog_tables_***_703-704.sql`
- `ftelog_tables_***_704-750.sql`
- `ftelog_tables_***_750-7502.sql`
- `ftelog_tables_***_7502-800.sql`

For example, if you are using IBM WebSphere MQ File Transfer Edition 7.0.3 and are migrating to IBM MQ 8.0.0.0, run 703-704, 704-750, 750-7502, and 7502-800.

Note:  On z/OS, you can go directly from 704 to 800, using `ftelog_tables_zos_704-800.sql`

The scripts up to `ftelog_tables_***_704-750.sql` are included in IBM MQ 8.0.0.0, and are located in the *MQ_INSTALLATION_PATH/mqft/sql* directory. Upgrades beyond 750 were not shipped with IBM MQ 8.0.0.0; if they are not present in the directory, you can download them from the link specified in [APAR IT01841](#).

6. Optional: If the IBM WebSphere MQ File Transfer Edition 7.0 database logger was configured as a Windows service, complete the following steps:
 - a) Reconfigure the database logger so that it is no longer a Windows service using the [fteModifyDatabaseLogger](#) command.
 - b) Reconfigure the Managed File Transfer in IBM WebSphere MQ 7.5 or later logger so that it is a Windows service using the [fteModifyLogger](#) command.
7. Start the Managed File Transfer in IBM WebSphere MQ 7.5 or later database logger.

Results

The database logger has now been migrated from IBM WebSphere MQ File Transfer Edition 7.0 to IBM WebSphere MQ 7.5, or later.

Increasing the page size of the log database on Db2 on Windows, UNIX or Linux

If your database is Db2 on a Windows, UNIX or Linux system, and you created your log database with a page size of less than 8 KB, you must increase the page size of the database before migrating to the IBM WebSphere MQ File Transfer Edition 7.0.3 or later tables.

Procedure

1. If you have not already stopped your database logger, stop your database logger using the **fteStopDatabaseLogger** command.
2. Back up your log database using the tools provided by Db2.
3. Use the Db2 **export** command to transfer the data from your log database tables to files on disk.
Note: You must specify large object files for tables that include large objects. Those tables are CALL_RESULT and METADATA.
4. Drop your log database tables.
5. Create a table space with a page size of at least 8 KB and with an associated buffer pool with a page size of at least 8 KB.
Give your new table space a name. For example, FTE8KSPACE.
6. Edit the `ftelog_tables_db2.sql` file so that the commands create tables in the new table space.
In the `ftelog_tables_db2.sql` file, change all occurrences of the text IN "USERSPACE1" to IN "*new_tablespace_name*". For example, change IN "USERSPACE1" to IN "FTE8KSPACE".
7. Run the SQL commands in the `ftelog_tables_db2.sql` file against your database.
8. Use the Db2 **load** command to transfer the exported data into the new tables.

Note:

- **Map the column names based on the column names found in the input file.** Ensure that the input column names and target column names match up in those tables that have changed their structure.
 - You must specify the IDENTITY OVERRIDE behavior on the identity column of all tables, except for MONITOR and TRANSFER. Specifying this behavior ensures that the row IDs are not regenerated during the load operation.
9. Run the Db2 **set integrity** command with integrity status values of **immediate** and **checked**, against the following tables in the order given:
 - CALL_ARGUMENT
 - MONITOR
 - MONITOR_ACTION
 - MONITOR_EXIT_RESULT
 - MONITOR_METADATA
 - SCHEDULE_ACTION
 - SCHEDULE
 - SCHEDULE_ITEM
 - TRANSFER
 - TRANSFER_CALLS
 - TRANSFER_EVENT
 - TRANSFER_ITEM
 - TRANSFER_STATS
 - TRIGGER_CONDITION

10. In tables with generated ID columns, set the ID generators to begin from a value one higher than the existing highest ID value.

The following tables have generated ID columns:

- AUTH_EVENT
- CALL
- CALL_ARGUMENT
- CALL_RESULT
- FILE_SPACE_ENTRY
- METADATA
- MONITOR_ACTION
- MONITOR_EXIT_RESULT
- MONITOR_METADATA
- SCHEDULE
- SCHEDULE_ACTION
- SCHEDULE_ITEM
- SCHEDULE_SPEC
- TRANSFER_CALLS
- TRANSFER_CD_NODE
- TRANSFER_CORRELATOR
- TRANSFER_EVENT
- TRANSFER_EXIT
- TRANSFER_ITEM
- TRANSFER_ITEM_ATTRIBUTES
- TRANSFER_STATS
- TRIGGER_CONDITION

To set the generated IDs of these tables to the correct value perform the following steps for each table:

a) Determine the maximum ID value in the existing data.

You can find this value by running this SQL statement:

```
SELECT MAX(ID) FROM FTELOG.table_name
```

The value returned from this command is the maximum existing ID in the specified table.

b) Alter the table to set the ID generator to begin from a new value that is 1 higher than the value returned by the previous step.

You can set this value by running the following SQL statement:

```
ALTER TABLE FTELOG.table_name ALTER COLUMN ID RESTART WITH value
```

Related tasks

[“Migrating from the stand-alone database logger to the Java EE database logger for MFT” on page 134](#)
You can migrate from the stand-alone database logger to the Java EE database logger. You must stop the stand-alone database logger and install the Java EE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration.

[“Migrating the database tables on Db2 on z/OS to IBM MQ 8.0” on page 21](#)

If your database is Db2 on a z/OS system, you must complete the following steps to migrate from IBM WebSphere MQ File Transfer Edition 7.0.3 to IBM WebSphere MQ File Transfer Edition 7.0.4, and from IBM WebSphere MQ File Transfer Edition 7.0.4 to Managed File Transfer in IBM MQ 8.0. The Db2 tables have different structures from previous releases. For example, there are new columns in some tables, and some variable characters columns can be larger, so the tables from previous releases have to be migrated to the Db2 format.

Migrating the database tables on Db2 on z/OS to IBM MQ 8.0

If your database is Db2 on a z/OS system, you must complete the following steps to migrate from IBM WebSphere MQ File Transfer Edition 7.0.3 to IBM WebSphere MQ File Transfer Edition 7.0.4, and from IBM WebSphere MQ File Transfer Edition 7.0.4 to Managed File Transfer in IBM MQ 8.0. The Db2 tables have different structures from previous releases. For example, there are new columns in some tables, and some variable characters columns can be larger, so the tables from previous releases have to be migrated to the Db2 format.

About this task

Managed File Transfer in IBM MQ 8.0 supports transferring very large files, where the size of the file is larger than can be stored in an integer (32 bit) number. There is a definition that uses BIGINT (64 bit) numbers. You can choose to use integer definitions, called `ftelog_tables_zos.sql`, or BIGINT definitions, called `ftelog_tables_zos_bigint.sql`, located in the `prod/mqf/sql` directory.

The BIGINT data type is available in IBM WebSphere MQ File Transfer Edition 7.0.4 Fix Pack 3, or Managed File Transfer in IBM WebSphere MQ 7.5.0 Fix Pack 1 or later.

To enable use of BIGINT data types, you must be using Db2 9.1 or later. INTEGER data types are used for fields which denote the sizes of files that are transferred and the table ID associated with each transfer. If you want to log transfers with file sizes greater than 2 GB, or if you want to store more than 2,147,483,648 individual transfers in your database you must use the BIGINT SQL file.

The following list outlines the processes you need to follow:

1. You have existing tables from IBM WebSphere MQ 7.0. These tables have a schema, for example FTELOG.
2. Create IBM MQ 8.0 tables using a different schema name, for example, FTEV8. This allows you to copy data from `FTELOG.table` to `FTE8.table`.
3. Copy the data to the new table
4. Set the generated ID values in the new tables
5. Run the **fteMigrateLogger** command to move the properties file to a new place in the directory structure.
6. Edit the logger properties file to specify the new schema (FTEV8)
7. Edit the existing Logger JCL to use the IBM MQ 8.0 Managed File Transfer libraries.
8. Start the logger.
9. Once the logger is working you can delete the FTELOG tables.

In the following description, the Managed File Transfer product is installed in the `/HMF8800` directory in USS.

Procedure

1. If you have not already stopped your database logger, stop your database logger using the **fteStopDatabaseLogger** command in USS or **P loggerjob**.
2. Issue the command `ls /HMF8800/mqft/sql` to list the SQL files in the directory.
If you are going to use BIGINT numbers copy, `ftelog_tables_zos_bigint.sql` to your home directory, otherwise, copy `ftelog_tables_zos.sql` to your home directory.
3. Edit the file you moved to your home directory:

- a) Change `fte1og` to the schema name for the new tables.
- b) Ensure each index has a unique name.
To do this, in an edit session:
 - i) Exclude all lines.
 - ii) Find `'CREATE UNIQUE INDEX ' ALL`
 - iii) Change `_KEY _K8Y ALL NX`
4. Check the file to make sure all of the statements are within column 71.
If the statements are not within column 71, split the line before column 71.
5. You might be able to use this file as input to SQL, or you might want to copy it to a PDS. To do this, edit the PDS and use the **copy** command, specifying the directory and file name.
6. Check the definitions with your Db2 administrator, as there might be site standards that you need to follow.
7. Carry out the following:
 - a) Copy the `_zos_704-800.sql` file, located in the `/HMF8800/mqft/sql/fte1og_tables` directory to your home directory.
 - b) Edit this file. Change `FTESRC` to your existing schema (`FTELOG`) and `FTEDEST` to the new schema (`FTEV8`).
 - c) Check the file to make sure all of the statements are within column 71.
If the statements are not within column 71, split the line before column 71.
 - d) If you have **DB2 RUNSTATS** jcl for the Managed File Transfer tables, create a new job specifying the new schema and tables.
8. Some tables have a generated ID to enforce a unique identifier for each row and you need to set these identifiers.
The following tables have generated ID columns:
 - `AUTH_EVENT`
 - `CALL`
 - `CALL_ARGUMENT`
 - `CALL_RESULT`
 - `FILE_SPACE_ENTRY`
 - `METADATA`
 - `MONITOR_ACTION`
 - `MONITOR_EXIT_RESULT`
 - `MONITOR_METADATA`
 - `SCHEDULE`
 - `SCHEDULE_ACTION`
 - `SCHEDULE_ITEM`
 - `SCHEDULE_SPEC`
 - `TRANSFER_CALLS`
 - `TRANSFER_CD_NODE`
 - `TRANSFER_CORRELATOR`
 - `TRANSFER_EVENT`
 - `TRANSFER_EXIT`
 - `TRANSFER_ITEM`
 - `TRANSFER_ITEM_ATTRIBUTES`
 - `TRANSFER_STATS`

- TRIGGER_CONDITION

To set the generated IDs of these tables to the correct value perform the following steps for each table:

- Determine the maximum ID value in the existing data.
You can find this value by running this SQL statement:

```
SELECT MAX(ID) FROM schema_name.table_name
```

The value returned from this command is the maximum existing ID in the specified table.

- Alter the table to set the ID generator to begin from a new value that is 1 higher than the value returned by the previous step.

You can set this value by running the following SQL statement:

```
ALTER TABLE schema_name.table_name ALTER COLUMN ID RESTART WITH value
```

- Edit the database properties file to specify the new schema name:
 - If your Managed File Transfer configuration directory is `/u/userid/fteconfig` you can use the USS command **`find /u/userid/fteconfig -name databaselogger.properties`** to locate the properties file for the logger.
 - Edit this file and change `wmqfte.database.schema` to the new schema value.
- Issue the following commands to convert the directory tree structure to 8.0.0 format before you attempt to use the logger:
 - `fteMigrateConfigurationOptions`**
 - `fteMigrateLogger`**
This copies the `databaselogger.properties` to `logger.properties`.
- Edit existing Logger JCL to use the 8.0.0 Managed File Transfer libraries.
- Start the logger.
Once the logger is working you can delete the V7 FTELOG tables.

Migrating Managed File Transfer for z/OS agents to a later version

Follow these instructions to migrate an IBM MQ Managed File Transfer for z/OS agent to a later version. This topic uses IBM MQ 8.0 as the initial version and IBM MQ 9.0 as the later version.

Important: These instructions assume the following points:

- The coordination queue manager and command queue manager for the Managed File Transfer for z/OS (MFT) topology are not changing during the migration.
- The agent queue manager is not being migrated. The agent queue manager remains at the same IBM MQ for z/OS level before and after the migration.
- The agent being migrated is administered using JCL.

Step 1 - Initial setup

- Install IBM MQ Managed File Transfer for z/OS 9.0 onto the same LPAR as IBM MQ Managed File Transfer for z/OS 8.0.
- Locate the SBFGCMDSDS PDSE for IBM MQ MFT in IBM MQ 9.0.
For more information about the SBFGCMDSDS PDSE, see [The SBFGCMDSDS library](#).
- Within this data set, edit the BFGCOPY member to specify the name of the PDSE that is to be used to administer the agent.
- Submit the BFGCOPY member. This copies the existing SBFGCMDSDS PDSE into a new PDSE with the name that was specified in Step “3” on page 23.

Step 2 - Modify the new PDSE

1. Within the new PDSE that you created using the BFGCOPY member, edit BFGCUSTM and update the following variables:

Variable	Value
++library++	Replace this variable with the name of the new PDSE.
armELEMENT	Set this to the same value that is used by the IBM MQ 8.0 agent.
armELEMENTYPE	Set this to the same value that is used by the IBM MQ 8.0 agent.
BFG_DATA	Set this to the same value that is used by the IBM MQ 8.0 agent.
BFG_GROUP_NAME	Set this to the same value that is used by the IBM MQ 8.0 agent.
BFG_JAVA_HOME	This needs to be set to the UNIX System Services (USS) path for the Java Runtime Environment that is to be used to run the agent. Set this to same value that is used by the IBM MQ 8.0 agent.
BFG_JVM_PROPERTIES	Set this to the same value that is used by the IBM MQ 8.0 agent.
BFG_PROD	This variable should be set to the USS path for the Managed File Transfer for z/OS in IBM MQ 9.0 installation.
BFG_WTO	Set this to the same value that is used by the IBM MQ 8.0 agent.
CLEAN_AGENT_PROPS	Set this to blank.
connectionQMgr	Set this to the same value that is used by the IBM MQ 8.0 agent.
coordinationQMgr	Set this to the same value that is used by the IBM MQ 8.0 agent.
CREDENTIALS_PATH	Set this to blank.
DB_PROPS_PATH	Set this to the same value that is used by the IBM MQ 8.0 agent.
DLQ_HLQ	Set this to the same value that is used by the IBM MQ 8.0 agent.
FTE_CONFIG	Set this to blank.
MQ_HLQ	Set this to the same value that is used by the IBM MQ 8.0 agent.
MQ_PATH	This variable needs to be set to the USS path for the IBM MQ installation associated with the agent queue manager.
NAME	Set this to the name of the agent that is being migrated.
QM	Set this to the same value that is used by the IBM MQ 8.0 agent. This ensures that when the agent is running IBM MQ 9.0, the agent uses the same agent queue manager that it did when running IBM MQ 8.0.
SERVICE_TYPE	AGENT
TMPDIR	Set this to the same value that is used by the IBM MQ 8.0 agent.

2. Save the modified BFGCUSTM member.
3. Submit BFGCUSTM. The member creates new members within the PDSE that can be used to administer the agent when it is using IBM MQ 9.0.



Attention: The member requires exclusive access to the PDSE, so you must stop using it while BFGCUSTM is running.

Step 3 - Perform the migration

1. Locate the IBM MQ 8.0 PDSE that is used to administer the agent.
2. Within this PDSE, submit the BFGAGSP member to stop the agent. The agent stops once it has completed all of its managed transfers.
3. Next, locate the new IBM MQ 9.0 PDSE that has just been created for the agent.
4. Inside this PDSE, submit the BFGAGST member. This starts the agent using IBM MQ Managed File Transfer for z/OS 9.0

Related concepts

[Using Managed File Transfer for z/OS](#)

You can apply the instructions in this topic as an alternative method of using Managed File Transfer in your enterprise, on your IBM MQ for z/OS system.

Migrating MFT to a new machine with a different operating system

The core steps required to successfully achieve a migration of MFT configurations to a new system or platform. The task is primarily focused on MFT configuration migration, but also discusses queue manager migration where appropriate.

Before you begin

Ensure that any agents you are going to migrate have completed any in-progress or pending transfers, and that you have taken a back up of:

- The coordination queue manager
- Agent queue managers
- Agents
- Resource Monitors
- Transfer Templates
- Scheduled Transfers

Important: IBM MQ installation names on one system are unlikely to match the installation names on the new system unless the old and new systems only have one installation, or you specify an installation name as part of the IBM MQ installation process.

About this task

The following migration procedure is based on the scenario where QMA is both the coordination queue manager for topology, and the agent queue manager for an agent called Agent1.

Agent1 has a monitor, transfer template and scheduled transfer. QMA also connects to a queue manager called QMB running on another system using their sender and receiver channels for file transfers.

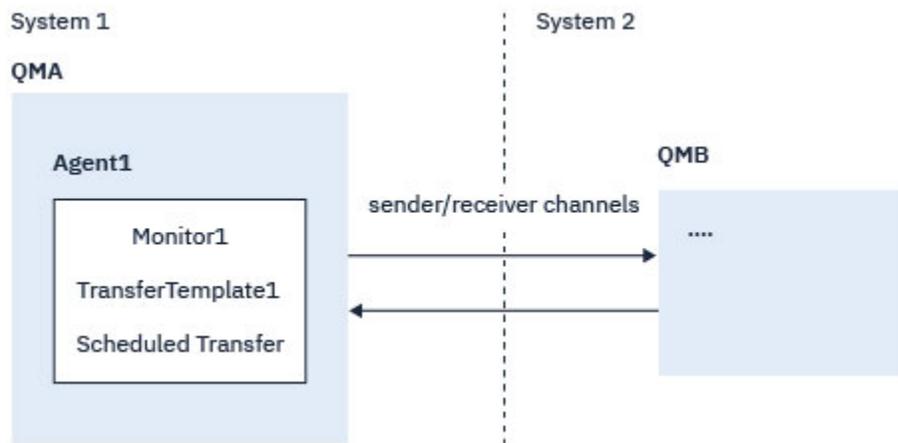


Figure 1. Migrating the MFT configuration on System 1



Attention: The following procedure explains only how to backup and restore MFT configurations. If you are migrating MFT to a new machine with the same operating system, queue manager data and log files can be backed up and restored by copying all the data files from the old system to the appropriate directories on the new system.

However, if the new machine has a different operating system, it is not possible to migrate the data files, because they are created platform specific.

Procedure

1. Backup procedure

- a) Save the queue manager configuration using the `dmpmqcfig` command to rebuild it later from its definition.

For example:

```
dmpmqcfig -m QMA -a > /mq/backups/QMA.mqsc
```

- b) Back up the configuration files for the agent that are stored under the IBM MQ data directory / MQ_DATA_PATH/mqft

The `mqft` directory normally has three sub-directories, which are `config`, `installation`, and `logs`. These contain agent installation data, configuration, and database logger files respectively.

If the agent is Protocol Bridge Agent, the `ProtocolBridgeCredentials.xml` file in the agent configuration directory also needs to be backed up. This file defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

- c) Export the configuration of the resource monitor to an XML file using the MFT `ftelistMonitors` command with the `-ox` option.

For example:

```
ftelistMonitors -ma Agent1 -mn Monitor -ox Monitor1Definition.xml
```

- d) Export transfer templates to XML files using the MFT `ftelistTemplates` command with the `-x` and `-o` options.

For example, the following command creates `TransferTemplate1.xml` in the current directory:

```
ftelistTemplates -x -o . TransferTemplate1
```

- e) Manually back up the scheduled transfer definitions.

It is not possible to export the definitions to XML files, but you can list scheduled transfers using the MFT `ftelistScheduledTransfers` command and backing up the definitions manually.

2. Recreate procedure

- a) Recreate queue manager QMA after installing IBM MQ and MFT on the new system.
- b) Restore the QMA configuration by running the `runmqsc` command to parse in the queue manager configuration saved in Step “1.a” on page 26

For example:

```
runmqsc QMA< /mq/backups/QMA.mqsc
```

- c) Recreate the sender and receiver channels that connect to QMB on System two.
- d) On the QMB queue manager side, update the connections details, such as host name and port number of the sender channel that connects to QMA.
- e) Recreate Agent1 by copying all of the backed up agent configuration files to the new system, and start the agent.
- f) Import the XML file for Monitor1 using the MFT `fteCreateMonitor` command with the `-ix` and `-f` options.

For example:

```
fteCreateMonitor -ix Monitor1Definition.xml -f
```

- g) Publish a message containing the contents of TransferTemplate1.xml in the message body to the SYSTEM.FTE topic on the coordination queue manager.

Use a stand-alone application, and specify the topic string:

```
SYSTEM.FTE/Templates/<template_id>
```

where `<template_id>` is the transfer template ID that can be found inside the `TransferTemplate1.xml` file.

For example, if the xml contains:

```
<?xml version="1.0" encoding="UTF-8"?><transferTemplateid="a7838085-0f2a-4980-b958-2dbbdfb22702" version="6.00">
```

, the topic string should be:

```
SYSTEM.FTE/Templates/a7838085-0f2a-4980-b958-2dbbdfb22702
```

- h) Recreate the scheduled transfers manually using the MFT `fteCreateTransfers` command.

MFT scenario overview

This section lists common Managed File Transfer topologies together with a scenario that sets up the system and transfers a test message.

- [Common topologies](#)
- [Configuring the base server](#)

MFT common topologies

This section lists common Managed File Transfer topologies. The double-sided arrows in each diagram represent connections to the queue manager.

See “[Connectivity considerations](#)” on page 31 for more information on queue manager connection options.

Base topology with one queue manager

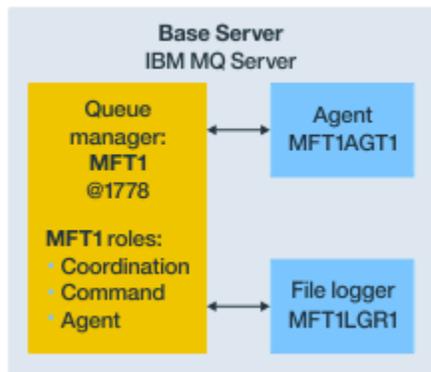


Figure 2. Base topology with one queue manager

A base topology represents a complete configuration which includes the coordination queue manager. The configuration name is the same as the name of the coordination queue manager. If the coordination queue manager name is MFT1, the configuration name is MFT1.

The base topology is the first Managed File Transfer configuration that you complete. After the base configuration is completed, partner agents from remote servers are added to the base configuration to exchange files.

The base topology does not exchange files outside the base topology server. However, the base topology enables you to move files to different locations in the same server and could be used for development purposes.

Base topology with one partner agent

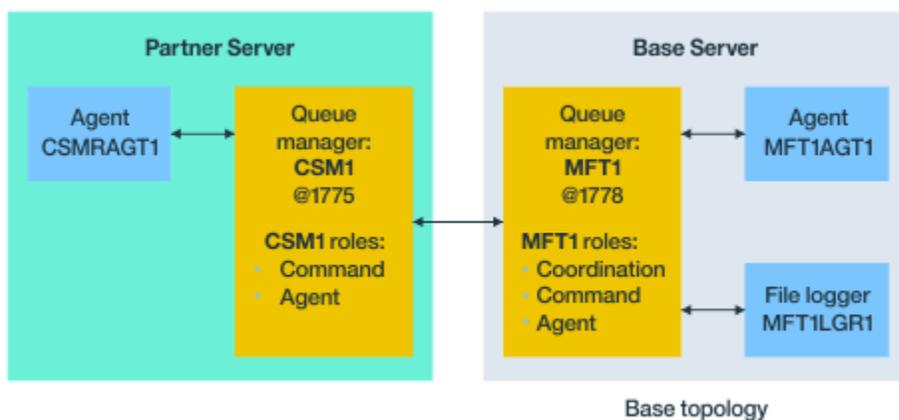


Figure 3. Base topology with one partner agent

This topology can exchange files between the two agents. Extra partner agents can be added in a similar way to the first added agent.

You can use a single queue manager for all three Managed File Transfer queue manager roles, or you can use dedicated queue managers for specific roles.

For example, you could have one queue manager dedicated to the coordination queue manager role, and the command and agent roles might share a second queue manager.

The connection between a remote agent queue manager in a separate server from the base configuration, and the base configuration coordination queue manager must be configured as an IBM MQ client, or MQI channel.

The connection to the coordination queue manager is established by the **fteSetupCoordination** command. If the coordination queue manager connection is not configured as an IBM MQ client channel, at the partner server, commands such as **fteListAgents** fail when issued from the partner agent server.

Base topology with separate coordination queue manager and one partner agent

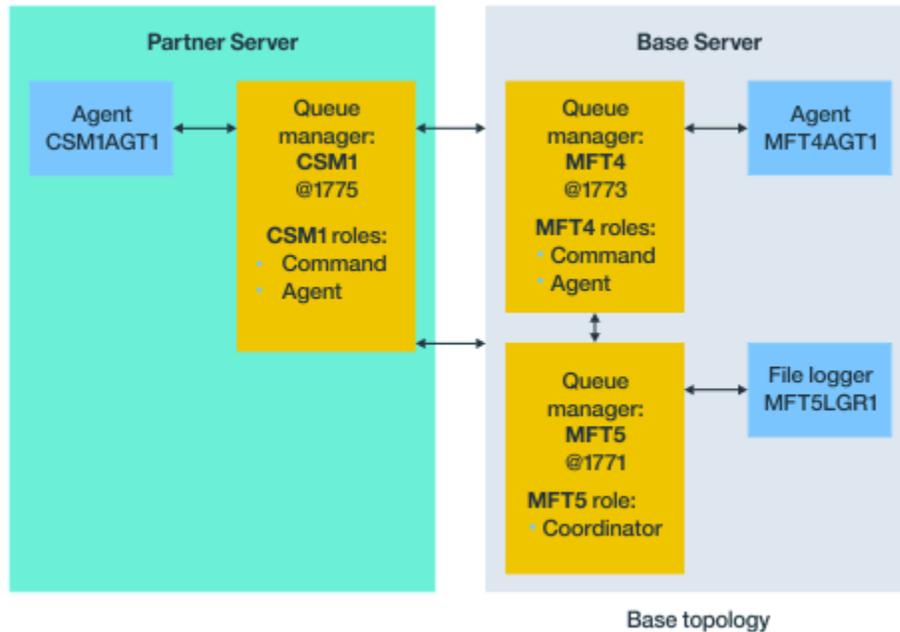


Figure 4. Base topology with separate coordination queue manager and one partner agent

In the base topology in Figure 3, on the base server, queue manager MFT4 is shared for the command and agent roles, and queue manager MFT5 is dedicated to the coordination queue manager role.

Connectivity must exist across all queue managers in the topology, including queue managers in the base topology, MFT4 and MFT5.

On the partner server queue manager, queue manager CSM1 has the roles of agent and commands queue manager.

This topology can exchange files between the two agents. Each partner agent must connect to a queue manager, as shown in the diagram. Extra partner agents can be added in a similar way, to the way that the first partner agent was added.

Base topology with Managed File Transfer Agent partner

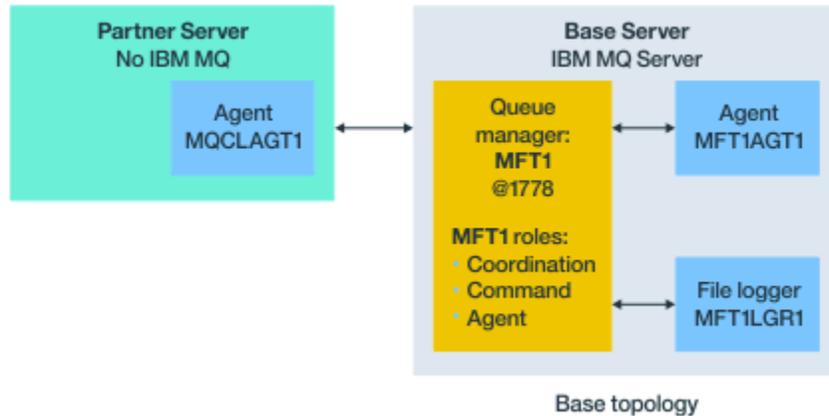


Figure 5. Base topology with Managed File Transfer Agent partner

This topology can exchange files between the two agents.

The server in the partner agent, depicted as MQCLAGT1 in the diagram, does not have IBM MQ server installed.

The partner agent is configured by using the same commands as the IBM MQ installed server, with some exceptions:

- The configuration for this partner agent must use IBM MQ client connections to base queue manager or queue managers.
- There is no need to run the coordination queue manager role IBM MQ definitions created by the configuration commands in the partner agent server. The coordination queue manager definitions already exist in the base server.

However, you must:

- Copy the agent object definitions generated when the agent is created in the partner server
- Transfer the definition file to the base configuration server, and
- Create the definitions in the queue manager identified as the agent queue manager in the base server.

In this case, MFT1 is serving all three roles, and you create the objects for agent MQCLAGT1 in the MFT1 queue manager.

As an alternative to copying the object definitions to the base server, you can run the **fteDefine** command for agent MQCLAGT1 on the base server where the agent queue manager is located. Use the definitions generated by the **fteDefine** command to create the required agent definitions on the agent queue manager.

For example, in the diagram shown, you would copy file MQCLAGT1_create.mqsc from the agent directory in the partner server, to the base configuration server, and create the required agent definitions in the MFT1 queue manager.

The configuration you complete on the partner agent server creates the Managed File Transfer configuration directory and required property files.

V 9.0.1 On the partner server, from IBM MQ 9.0.1 onwards, you can install the Managed File Transfer Redistributable Client from Fix Central.

Note: The MQMFT redistributable client differs from the IBM MQ redistributable client, in that it is already packaged and does not require use of the **genmqpkg** utility. For more information, see [Redistributable Clients](#).

Connectivity considerations

In the preceding diagrams, each line across the agents and queue managers represents a connection to a queue manager.

This connection might be:

- A local connection
- A bindings or message channel connection, or
- An IBM MQ client or MQI connection.

The type of connection you select in your configuration depends on the parameters you specify

- When you specify the queue manager name parameter without other connection parameters, you specify a bindings connection.

If the queue manager used is local to the Managed File Transfer configuration, it also represents a local connection, when used in the base configuration server.

- If you specify the queue manager name parameter, along with the corresponding host, port, and channel name parameters, you specify an IBM MQ client connection.

When agents are located on the same host as the agent queue manager, a bindings type specification, which results in a local connection, is more efficient.

Configuring the base server

How you set up the base server with a separate configuration queue manager.

Before you begin

The following example assumes that you have:

- Reviewed the section [“Connectivity considerations” on page 31](#) and understand how to influence the type of connection to queue managers in the configuration.
- A working IBM MQ infrastructure. See [Configuring IBM MQ queue managers](#) for information on setting up queue managers.
- IBM MQ security tasks are completed.

All system resources, such as access to files, are configured with adequate security.

For Managed File Transfer security configuration, refer to [Security overview for Managed File Transfer and User authorities on Managed File Transfer actions](#).

- All IBM MQ connections are tested after IBM MQ is configured by either using a sample program to send and receive messages, or using sample **amqscnxc** to test IBM MQ client type connections.

The **amqscnxc** sample connects to a queue manager by defining the channel connection in the sample code, which is similar to the way that Managed File Transfer connects, when it uses an MQI or IBM MQ client type connection.

- The instructions assume that the server you use for the base configuration has one IBM MQ version installed. If you have multiple IBM MQ installations in the base server, you must be careful to use the correct file path for the version of IBM MQ you want to use.
- The queue managers used in these instructions do not require connection authentication.

While it might be simpler to complete your first configuration without connection authentication required, if your enterprise requires immediate use of connection authentication, see [Managed File Transfer and IBM MQ connection authentication](#) for instructions on how to configure an `MQMFTCcredentials.xml` credentials file

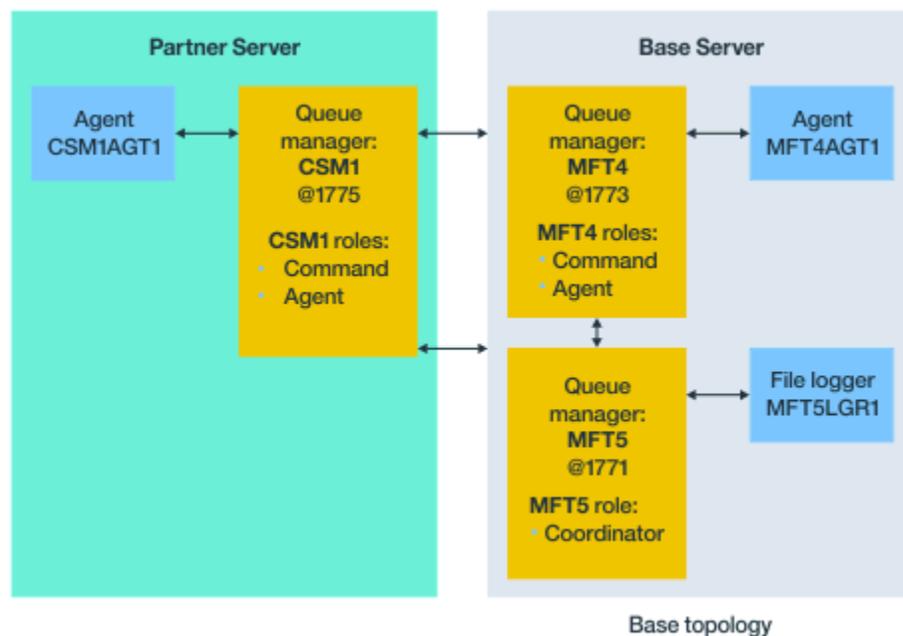


Figure 6. Base topology with separate coordination queue manager and one partner agent

About this task

The queue manager roles for the example configuration are:

- Base server
 - Queue manager MFT5 is the coordination queue manager
 - Queue manager MFT4 is used as the agent queue manager for agent MFT4AGT1, and also serves as the command queue manager for the MFT5 configuration on the base server.
- Partner server
 - Queue manager CSM1 doubles as the agent queue manager for agent CSM1AGT1, and as the command queue manager for the MFT5 configuration on the partner server.
 - Queue manager MFT5, on the base server, is the coordination queue manager.

Procedure

1. [Configure the coordination queue manager](#)
2. [Configure the command queue manager](#)
3. [Set up the agent](#)
4. [Set up the logger](#)
5. [Configure a partner server](#)

What to do next

Set up the [MQExplorer](#) with [MQMFT](#) so that you can test your example setup.

Configuring the coordination queue manager

How you configure the Coordination queue manager to coordinate file transfers.

Before you begin

Ensure that you have full connectivity between the queue managers you set up for this scenario.

About this task

This task sets up the coordination queue manager MFT5, and the instructions in this section assume that you are working with one IBM MQ installation.

If you have multiple installations, you must set the IBM MQ path to the version of IBM MQ required, by using the `setmqenv` command, before you start any of the configuration tasks.

Procedure

1. Log in as the Managed File Transfer administrator.
2. Issue the following command to identify the coordination queue manager and set up the configuration directory structure:

```
fteSetupCoordination -coordinationQMgr MFT5
```

Coordination queue manager directory

```
C:\data\mqft\config\MFT5
```

coordination.properties file

```
C:\data\mqft\config\MFT5\coordination.properties
```

The command also produces an MQSC command file that you must run against your coordination queue manager C:\data\mqft\config\MFT5\MFT5.mqsc:

3. Change to the C:\data\mqft\config\MFT5 directory.
4. Configure the queue manager, to act as the coordination queue manager, by running the following command.

You need to provide the MQSC command file, produced by the command you issued in Step “2” on [page 33](#):

```
runmqsc MFT5 < MFT5.mqsc > mft5.txt
```

5. Open the `mft5.txt` results file with your preferred editor. and ensure that the definitions have been created successfully.

What to do next

Set up the [command queue manager](#).

Configuring the commands queue manager

How you configure the commands queue manager.

Before you begin

Ensure that you have configured the coordination queue manager. See [“Configuring the coordination queue manager”](#) on [page 32](#) for more information.

About this task

This task identifies the commands queue manager.

Procedure

Issue the following command:

```
fteSetupCommands -connectionQMgr MFT4
```

You obtain the following message BFGCL0245I: The file
C:\data\mqft\config\MFT4\command.properties has been created successfully.

The command queue manager does not require extran IBM MQ definitions. After you run **fteSetupCommands**, the command.properties file is created in the MFT5 configuration directory.

What to do next

Set up the [agent](#).

Setting up the agent

How you prepare a file transfer agent MFT4AGT1, including MQSC scripts that you must run.

Before you begin

You should have set up the command queue manager. See [“Configuring the commands queue manager”](#) on page 33 for more information.

About this task

This task prepares the Windows file transfer agent, MFT4AGT1.

Procedure

1. Issue the following command:

```
fteCreateAgent -agentName MFT4AGT1 -agentQMgr MFT4
```

After you create the agent with the **fteCreateAgent** command, the agents directory, and a subdirectory for the agent, MFT4AGT1, are added to the MFT5 directory.

In the `data\MFT5\agents\MFT4AGT1` directory you find the:

- agent.properties file
 - MFT4AGT1_create.mqsc file, which contains IBM MQ definitions required by the agent.
2. Change to the `data\MFT5\agents\MFT4AGT1` directory, and create the required agent queue manager definitions by issuing the following command:

```
runmqsc MFT4 < MFT4AGT1_create.mqsc > mft4.txt
```

3. Open the `mft4.txt` results file with your preferred editor and ensure that the definitions have been created successfully.
4. Start the agent by typing the following command: **fteStartAgent** MFT4AGT1.
5. Display the agent by typing the following command: **fteListAgents**.

You should see output similar to the following:

```
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGPR0127W: No credentials file has been specified to connect to IBM MQ.
Therefore, the assumption is that IBM MQ authentication has been disabled.
Agent Name:      Queue Manager Name:      Status:
MFT4AGT1        MFT4                        READY
```

Note: if you have not enabled connection authentication in your Managed File Transfer environment, you can ignore the BFGPR0127W message.

If you issue the **fteListAgents** command and receive the following message, BFGCL0014W: No agents exist that match the current selection criteria., see [“What to do if your MFT agent is not listed by the fteListAgents command”](#) on page 289 for further information.

What to do next

Set up the [logger](#).

Setting up the logger

A file or database logger is required to keep history and audit information about transfer activity for the configuration. In this example you create a file logger.

Before you begin

You must have set up the:

- Configuration queue manager
- Command queue manager
- Agent

Procedure

1. Issue the following command:

```
fteCreateLogger -loggerQMgr MFT5 -loggerType FILE  
-fileLoggerMode CIRCULAR -fileSize 5MB -fileCount 3 MFT5lgr1
```

After you run the **fteCreateLogger** command, the `data\mqft\config\MFT5\loggers` directory is created, with an MFT5LGR1 subdirectory.

The MFT5LGR1 subdirectory holds the `logger.properties` file. Also in the directory is a file called `MFT5LGR1_create.mqsc` with IBM MQ definitions required by the logger.

2. Change to the directory `data\mqft\config\MFT5\loggers\MFT5LGR1`.
3. Run the associated MQSC command file.

```
runmqsc MFT5 < MFT5_create.mqsc
```

to create the definitions required by the logger.

- a) Review the results of the object definitions to confirm that the required objects have been created successfully.
4. Start the logger by issuing the following command **fteStartLogger** MFT5LGR1.
 5. Review the contents of file `output0.log` at `data\mqft\logs\MFT5\loggers\MFT5LGR1\logs`.

After some information about the logger, the last statement should contain message: BFGDB0023I: The logger has completed startup activities and is now running.

Occasionally, log information might not be written to the `output0.log` the first time the logger starts. If the `output0.log` file is empty, restart the logger by typing **fteStopLogger** MFT5LGR1 and pressing the **Enter** key.

Restart the logger by typing **fteStartLogger** MFT5LGR1 and pressing the **Enter** key. File `output0.log` now shows data.

The same behavior extends to the agent version of the `output0.log` file the first time an agent is started.

Stop and start the agent by using **fteStopAgent** and **fteStartAgent** commands. You then see log data written to the agent `output0.log` file.

Results

You have configured the base server, which includes the coordination queue manager for this configuration.

What to do next

You now do similar work for the partner server, which contains a remote agent.

Configuring a partner server

How you configure a partner server, when the base server has a separate coordination queue manager

Before you begin

Ensure that you have fully completed all the tasks to set up a base server, that includes a configuration queue manager.

About this task

The same assumptions made about IBM MQ and the security configuration, as well as the IBM MQ path also apply to the partner server.

Start by setting up the MFT5 configuration directory, and identifying the coordination queue manager by using the **fteSetupCoordination** command.

Procedure

1. Create the partner server configuration directory by issuing the following command:

```
fteSetupCoordination -coordinationQMgr MFT5  
-coordinationQMgrHost 177.16.20.15 -coordinationQMgrPort 1771  
-coordinationQMgrChannel MQMFT.MFT5.SVRCONN
```

Notes:

- a. When the coordination queue manager is on a different server from the partner server, the connection to the base server coordination queue manager must be defined as a client connection.

Failure to define the coordination queue manager connection as an IBM MQ client connection, on the partner server, causes any Managed File Transfer command, that connects to the coordination queue manager, to fail.

An example of a command that connects to the coordination queue manager is **fteListAgents**.
 - b. You do not need to create the IBM MQ definitions as the definitions required by the coordination queue manager were completed when you configured the base server.
2. Identify the commands queue manager by issuing the following command:

```
fteSetupCommands -connectionQMgr CSM1
```

The commands queue manager does not require any extran IBM MQ definitions.

3. Identify the partner agent queue manager, and create the partner agent queue manager, by issuing the following command:

```
fteCreateAgent -agentName CSM1AGT1 -agentQMgr CSM1
```

4. Change to the CSM1AGT1 directory.
5. Create the IBM MQ definitions required by the agent, by issuing the following command:

```
runmqsc CSM1 < CSM1AGT1_create.mqsc > csm1.txt
```

- a) Open file `csm1.txt` with your preferred editor to confirm that all agent required definitions have been created successfully.
6. Start the agent, by issuing the following command:

```
fteStartAgent CSM1AGT1
```

7. Display the agent by typing **fteListAgents**

You should see output similar to the following:

```
C:\>fteListAgents
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGPR0127W: No credentials file has been specified to connect to IBM MQ. Therefo
re, the assumption is that IBM MQ authentication has been disabled.
Agent Name:      Queue Manager Name:  Status:
CSM1AGT1        CSM1                   READY
MFT4AGT1        MFT4                   READY
```

Note: if you have not enabled connection authentication in your Managed File Transfer environment, you can ignore the BFGPR0127W message.

If you issue the **ftelistAgents** command and receive the following message, BFGCL0014W: No agents exist that match the current selection criteria., see [“What to do if your MFT agent is not listed by the fteListAgents command”](#) on page 289 for further information.

If the status of one of the agents is UNREACHABLE, see [“What to do if an agent is shown as being in an UNKNOWN state”](#) on page 290 for further information.

Setting up the IBM MQ Explorer with MFT

This task helps you connect IBM MQ Explorer to the Managed File Transfer configuration.

Procedure

1. Start IBM MQ Explorer.
2. In the left Navigator panel, scroll down and expand the folder: Managed File Transfer.
You see the entry for the Coordination queue manager: MFT5
3. Right click on MFT5 and select **Connect**.
 - a) Select Agents in the drop down menu that appears and ensure that both agents, MFT4AGT1 and CSMAGT1, are in the Ready state.

What to do next

Test your example setup with [IBM MQ Explorer](#).

Using the IBM MQ Explorer to test a file transfer

This task gives an example of how you use IBM MQ Explorer with Managed File Transfer, to test a file transfer, after you have set up the IBM MQ Explorer as described in the previous topic.

Before you begin

Ensure that you have a working system, that the agents are READY and IBM MQ Explorer is working. See [“Setting up the IBM MQ Explorer with MFT”](#) on page 37 for more information.

About this task

Determine the file to use to test the transfer, and a directory to copy it to. For this example, it is assumed that file `test-file.txt` out of directory `C:\temp\mft` is used.

```
C:\temp\mft> dir *
Date stamp 61 test-file.txt
1 File(s) 61 bytes
```

Procedure

1. Start the IBM MQ Explorer in Windows
2. In the left Navigator panel, expand the folder: Managed File Transfer.
You see the entry for the Coordination queue manager: MFT5
3. Right click on MFT5 and select **Connect**.
4. Once connected, right click on MFT5 and select **New Transfer**
 - a) Use the pull down menu to select MFT4AGT1 for the Source agent and CSMAGT1 for the Destination agent.
 - b) Click **Next**.
 - c) Click **Add** on the next window.
A wide dialog appears. The left side is for Source and the right side for Destination.
5. On the Source panel:
 - a) Select **Text transfer** as the file is text.
 - b) Select **Browse** to locate the file.
In this case, the file is C:\temp\mft\test-file.txt.

 **Attention:** Do not click **OK** as you need to complete the Destination panel.
6. On the Destination panel:
 - a) Enter the name that you are giving the file at the destination, for example, test-file.txt.
The use of relative paths is supported. The top portion of the full path is the home directory of the user ID who starts the destination agent.
 - b) Select **Overwrite files if present** if you require this option.
 - c) Click **OK**.
The file you selected appears in the **New Transfers** panel.
7. If the MFT5 configuration menu is closed, and shows +MFT5, expand the menu by clicking the + sign.
8. Stay at the MFT configuration selected.
Next, you check the status of the transfer by carrying out the following procedure.
9. Click **Transfer Log** under the coordination queue manager MFT5.
10. Look at the status in Managed File Transfer - Current Transfer progress panel, immediately below the **Transfer Log** top panel and wait for the transfer to complete.
If the transfer shows successful and with a green background, you have successfully completed the test of your configuration.
If the transfer failed with a red background, an error occurred.
In most cases, you can use the scroll bar below the top **Transfer Log** panel and view a summary of the reasons for failure.
 - a) If you cannot determine why the transfer failed, double-click the entry for the transfer in the top **Transfer Log** panel.
 - b) Select XML in the left pane of the pop-up panel that appears.
 - c) Scroll through the information to determine the cause of the error.
 - d) Make the corrections needed and test the transfer again.

Securing MFT

Directly after installation and with no modification, Managed File Transfer has a level of security that might be suitable for test or evaluation purposes in a protected environment. However, in a production environment, you must consider appropriately controlling who can start file transfer operations, who can read and write the files being transferred, and how to protect the integrity of files.

Related concepts

[“MFT sandboxes” on page 45](#)

You can restrict the area of the file system that the agent can access as part of a transfer. The area that the agent is restricted to is called the sandbox. You can apply restrictions to either the agent or to the user that requests a transfer.

Related tasks

[“Configuring SSL or TLS encryption for MFT” on page 51](#)

You can use SSL or TLS can be used with IBM MQ Managed File Transfer to secure the communication between agents and their agent queue managers, commands and the queue managers that they are connecting to, and the various queue manager to queue manager connections within your topology.

Related reference

[“Restricting group authorities for MFT-specific resources” on page 337](#)

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

[“Managing authorities for MFT-specific resources” on page 335](#)

For any file transfer request, the Managed File Transfer Agent processes require some level of access to their local file systems. In addition, both the user identifier associated with the agent process, and the user identifiers associated with users performing file transfer operations must have the authority to use certain IBM MQ objects.

[“Authorities for MFT to access file systems” on page 347](#)

For any file transfer request, the Managed File Transfer agent processes require some level of access to their local file systems.

[“MFT and IBM MQ connection authentication” on page 39](#)

Connection authentication allows a queue manager to be configured to authenticate applications by using a provided user ID and password. If the associated queue manager has security enabled, and requires credential details (user ID and password), the connection authentication feature must be enabled before a successful connection to a queue manager can be made. Connection authentication can be run in compatibility mode or MQCSP authentication mode.

[“commandPath MFT property” on page 347](#)

Use the commandPath property to specify the locations that Managed File Transfer can run commands from. Take extreme care when you set this property because any command in one of the specified commandPaths can effectively be called from a remote client system that is able to send commands to the agent.

[“Authority to publish MFT Agents log and status messages” on page 346](#)

Managed File Transfer Agents issue various log, progress, and status messages that are published on the coordination queue manager. The publication of these messages is subject to the IBM MQ security model, and in some cases you might have to perform further configuration to enable publication.

MFT and IBM MQ connection authentication

Connection authentication allows a queue manager to be configured to authenticate applications by using a provided user ID and password. If the associated queue manager has security enabled, and requires credential details (user ID and password), the connection authentication feature must be enabled before a successful connection to a queue manager can be made. Connection authentication can be run in compatibility mode or MQCSP authentication mode.

Methods of supplying credential details

Many Managed File Transfer commands support the following methods of supplying credential details:

Details supplied by command line arguments.

The credential details can be specified by using the **-mquserid** and **-mqpassword** parameters. If the **-mqpassword** is not supplied, then the user is asked for the password where the input is not displayed.

Details supplied from a credentials file: MQMFTCredentials.xml.

The credential details can be predefined in a MQMFTCredentials.xml file either as clear text or obfuscated text.

For information about setting up an MQMFTCredentials.xml file on IBM MQ for Multiplatforms see [“Configuring MQMFTCredentials.xml on multiplatforms”](#) on page 40.

For information about setting up an MQMFTCredentials.xml file on IBM MQ for z/OS see [“Configuring MQMFTCredentials.xml on z/OS”](#) on page 42.

Precedence

The precedence of determining the credential details is:

1. Command line argument.
2. MQMFTCredentials.xml index by associated queue manager and user running the command.
3. MQMFTCredentials.xml index by associated queue manager.
4. Default backward compatibility mode where no credential details are supplied to allow compatibility with previous releases of IBM MQ or IBM WebSphere MQ

Notes:

- The **fteStartAgent** and **fteStartLogger** commands do not support the command line argument **-mquserid**, or **-mqpassword**, and the credential details can only be specified with the MQMFTCredentials.xml file.

• z/OS

On z/OS, the password must be uppercase, even if the user's password has lowercase letters. For example, if the user's password was "password", it would have to be entered as "PASSWORD".

Related reference

[Which MFT command connects to which queue manager](#)

A Managed File Transfer topology consists of a number of different components.

[MFT credentials file format](#)

The MQMFTCredentials.xml file contains sensitive user ID and password information. The elements in the MQMFTCredentials.xml file must conform to the MQMFTCredentials.xsd schema. The security of credentials files is the responsibility of the user.

Configuring MQMFTCredentials.xml on multiplatforms

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT commands that connect with a queue manager to supply user ID and password credentials. Similarly, MFT loggers might be required to specify a user ID and password when connecting to a database. This credential information can be stored in the MFT credentials file.

About this task

The elements in the MQMFTCredentials.xml file must conform to the MQMFTCredentials.xsd schema. For information about the format of MQMFTCredentials.xml, see [MFT credentials file format](#).

You can find a sample credentials file in the MQ_INSTALLATION_PATH/mqft/samples/credentials directory.

You can have one MFT credentials file for the coordination queue manager, one for the command queue manager, one for each agent, and one for each logger. Alternatively, you can have one file which is used by everything in your topology.

The default location of the MFT credentials file is as follows:

Linux **UNIX and Linux**

\$HOME

Windows **Windows**

%USERPROFILE% or %HOMEDRIVE%%HOMEPATH%

If the credentials file is stored in a different location, then you can use the following properties to specify where the commands should look for it:

Table 1. : Properties that define the location of the MQMFTCredentials.xml file for various commands.

Type of command	Property file	Property name
Command which connects to the coordination queue manager	coordination.properties	coordinationQMgrAuthenticationCredentialsFile
Command which connects to the command queue manager	connection.properties	connectionQMgrAuthenticationCredentialsFile
Command that connects to an agent process	agent.properties	agentQMgrAuthenticationCredentialsFile
Command that connects to a logger process	logger.properties	loggerQMgrAuthenticationCredentialsFile

Table 2. : Properties that define the location of the MQMFTCredentials.xml file for agents and logger processes.

Type of command	Property file	Property name
MFT agents	agent.properties	agentQMgrAuthenticationCredentialsFile
MFT loggers	logger.properties	loggerQMgrAuthenticationCredentialsFile

For details about what commands and processes connect to which queue manager, see [Which MFT commands and processes connect to which queue manager](#).

Because the credentials file contains user ID and password information, it requires special permissions to prevent unauthorized access to it:

Linux **UNIX and Linux**

```
chown <agent owner userid>
chmod 600
```

Windows **Windows**

Ensure that inheritance is not enabled, and then remove all of the user IDs except those running the agent or logger that will be using the credentials file.

The credential details used to connect to an MFT coordination queue manager, in the IBM MQ Explorer Managed File Transfer plug-in for , depends on the type of configuration:

Global (configuration on local disk)

A global configuration uses the credentials file specified in the coordination and command properties.

Local (defined within IBM MQ Explorer):

A local configuration uses the properties of the connection details of the associated queue manager in IBM MQ Explorer.

Related tasks

[“Enabling MQCSP authentication” on page 44](#)

From IBM MQ 9.0.3 for Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 2 for Long Term Support, you can enable MQCSP authentication mode for connection authentication of the IBM MQ Explorer MFT Plugin connecting with a coordination queue manager or command queue manager. You can also enable MQCSP authentication mode for connection authentication for a Managed File Transfer agent connecting with a coordination queue manager or command queue manager.

Related reference

[MFT credentials file format](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

[fte0bfuscate](#): encrypt sensitive data

Configuring MQMFTCredentials.xml on z/OS

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT agents, and commands that connect to a queue manager, to supply user ID and password credentials.

Similarly, MFT loggers might be required to specify a user ID and password when connecting to a database.

This credential information can be stored in the MFT credentials file. Note that the credentials files are optional, however, it is easier to define the file or files that you require before you customize the environment.

In addition to this, if you have credentials files, you receive fewer warning messages. The warning messages inform you that MFT considers that queue manager security is off, and therefore you are not supplying authentication details.

You can find a sample credentials file in the `MQ_INSTALLATION_PATH/mqft/samples/credentials` directory.

Here is an example of an `MQMFTCredentials.xml` file:

```
<?xml version="1.0" encoding="IBM-1047"?>
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/MFTCredentials"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wmqfte.ibm.com/MFTCredentials MFTCredentials.xsd">
  <tns:qmgr name="MQPH" user="ADMIN" mqUserId="JOHNDOEH" mqPassword="cXXXX" />
  <tns:qmgr name="MQPI" user="ADMIN" mqUserId="JOHNDOEI" mqPassword="yXXXX" />
  <tns:qmgr name="MQPH" mqUserId="NONEH" mqPassword="yXXXX" />
  <tns:qmgr name="MQPI" mqUserId="NONEI" mqPassword="yXXXX" />
</tns:mqmftCredentials>
```

When a job with userid ADMIN needs to connect to queue manager MQPH, it passes user ID *JOHNDOEH* and uses password *cXXXX*.

If the job is run by any other user ID, and connects MQPH, that job passes user ID *NONEH* and password *yXXXX*.

The default location for the `MQMFTCredentials.xml` file is the user's home directory on z/OS Unix System Services (USS). It is also possible to store the file in either a different location on USS, or in a member within a partitioned data set.

If the credentials file is stored in a different location, then you can use the following properties to specify where the commands should look for it:

Table 3. : Properties that define the location of the MQMFTCredentials.xml file for various commands.

Type of command	Property file	Property name
Command which connects to the coordination queue manager	coordination.properties	coordinationQMGrAuthenticationCredentialsFile
Command which connects to the command queue manager	connection.properties	connectionQMGrAuthenticationCredentialsFile
Command that connects to an agent process	agent.properties	agentQMGrAuthenticationCredentialsFile
Command that connects to a logger process	logger.properties	loggerQMGrAuthenticationCredentialsFile

Table 4. : Properties that define the location of the MQMFTCredentials.xml file for agents and logger processes.

Type of command	Property file	Property name
MFT agents	agent.properties	agentQMGrAuthenticationCredentialsFile
MFT loggers	logger.properties	loggerQMGrAuthenticationCredentialsFile

For details about what commands and processes connect to which queue manager, see [Which MFT commands and processes connect to which queue manager](#).

To create the credentials file within a partitioned data set, carry out the following steps:

- Create a PDSE with format VB and logical record length (Lrecl) 200.
- Create a member within the data set, make a note of the data set and member, and add the following code to the member:

```
<?xml version="1.0" encoding="IBM-1047"?>
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wmqfte.ibm.com/MFTCredentials MQMFTCredentials.xsd">
  <!--credentials information goes here-->
</tns:mqmftCredentials>
```

You can protect the credentials file using a security product, for example, RACF®, but the user IDs running the Managed File Transfer commands, and administering the agent and logger processes, need read access to this file.

You can obscure information in this file using the JCL in member BFGCROBS. This takes the file and encrypts the IBM MQ user ID and password. For example member BFGCROBS takes the line

```
<tns:qmgr name="MQPI" user="JOHND0E2" mqUserId="JOHND0E1" mqPassword="yXXXX" />
```

and creates

```
<tns:qmgr mqPasswordCipher="e977c61e9b9c363c" mqUserIdCipher="c394c5887867157c"
name="MQPI" user="JOHND0E2"/>
```

If you want to keep the user ID to IBM MQ user ID mapping, you can add comments to the file. For example

```
<!-- name="MQPI" user="ADMIN" mqUserId="JOHND0E1 -->
```

These comments are unchanged by the obscuring process.

Note that the content is obscured, not strongly encrypted. You should limit which user IDs have access to the file.

Related tasks

[“Configuring MQMFTCredentials.xml on multiplatforms” on page 40](#)

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT commands that connect with a queue manager to supply user ID and password credentials. Similarly, MFT loggers might be required to specify a user ID and password when connecting to a database. This credential information can be stored in the MFT credentials file.

V 9.0.0.2 V 9.0.3 Enabling MQCSP authentication

From IBM MQ 9.0.3 for Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 2 for Long Term Support, you can enable MQCSP authentication mode for connection authentication of the IBM MQ Explorer MFT Plugin connecting with a coordination queue manager or command queue manager. You can also enable MQCSP authentication mode for connection authentication for a Managed File Transfer agent connecting with a coordination queue manager or command queue manager.

About this task

If you use the IBM MQ Explorer Managed File Transfer plugin, or have Managed File Transfer agents that connect to a queue manager using the CLIENT transport and specify a password, then the agent does not authenticate with the queue manager if the password specified is greater than 12 characters in length. This is because the code does not use MQCSP authentication, and authenticates using compatibility mode, which limits the password to 12 characters in length.

From IBM MQ 9.0.3 and IBM MQ 9.0.0 Fix Pack 2, you can disable the default compatibility mode and enable MQCSP authentication mode.

Procedure

- To disable compatibility mode and enable MQCSP authentication for a coordination queue manager or command queue manager in IBM MQ Explorer, complete the following steps:
 - a) Select the queue manager that you want to connect to.
 - b) Right click, and select **Connection Details->Properties** from the pop-up menu.
 - c) Click the **Userid** tab.
 - d) Ensure that **Enable user identification** is selected, and clear the **User identification compatibility mode** check box.
- To disable compatibility mode and enable MQCSP authentication for a Managed File Transfer agent, add the parameter **useMQCSPAuthentication** to the `MQMFTCredentials.xml` file for the relevant user and set it to `true`.

The parameter must be set to `true`. If the parameter is not specified, it is by default set to `false` and compatibility mode is used to authenticate the user with the queue manager.

The following example shows how to set the **useMQCSPAuthentication** parameter in the `MQMFTCredentials.xml` file:

```
<tns:qmgr name="CoordQueueMgr" user="ernest" mqUserId="ernest"
mqPassword="AveryL0ngPassw0rd2135" useMQCSPAuthentication="true"/>
```

Related concepts

[MQCSP password protection](#)

Related reference

[“MFT and IBM MQ connection authentication” on page 39](#)

Connection authentication allows a queue manager to be configured to authenticate applications by using a provided user ID and password. If the associated queue manager has security enabled, and requires credential details (user ID and password), the connection authentication feature must be enabled before

a successful connection to a queue manager can be made. Connection authentication can be run in compatibility mode or MQCSP authentication mode.

[“MFT credentials file format” on page 785](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

MFT sandboxes

You can restrict the area of the file system that the agent can access as part of a transfer. The area that the agent is restricted to is called the sandbox. You can apply restrictions to either the agent or to the user that requests a transfer.

Sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent. You can not use agent sandboxing for agents that need to transfer to or from IBM MQ queues.

Related reference

[“Working with MFT agent sandboxes” on page 45](#)

To add an additional level of security to Managed File Transfer, you can restrict the area of a file system that an agent can access.

[“Working with MFT user sandboxes” on page 46](#)

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

Working with MFT agent sandboxes

To add an additional level of security to Managed File Transfer, you can restrict the area of a file system that an agent can access.

You cannot use agent sandboxing for agents that transfer to or from IBM MQ queues. Restricting access to IBM MQ queues with sandboxing can be implemented instead by using user sandboxing which is the recommended solution for any sandboxing requirements. For more information about user sandboxing, see [“Working with MFT user sandboxes” on page 46](#)

To enable agent sandboxing, add the following property to the `agent.properties` file for the agent you want to restrict:

```
sandboxRoot=[!]restricted_directory_nameseparator...separator[!]restricted_directory_name
```

where:

- `restricted_directory_name` is a directory path to be allowed or denied.
- `!` is optional and specifies that the following value for `restricted_directory_name` is denied (excluded). If `!` is not specified `restricted_directory_name` is an allowed (included) path.
- `separator` is the platform-specific separator.

For example, if you want to restrict the access that AGENT1 has to the `/tmp` directory only, but not allow the subdirectory `private` to be accessed, set the property as follows in the `agent.properties` file belonging to AGENT1: `sandboxRoot=/tmp:!/tmp/private`.

The `sandboxRoot` property is described in [Advanced agent properties](#).

Both agent and user sandboxing are not supported on protocol bridge agents or on Connect:Direct bridge agents.

Working in a sandbox on UNIX, Linux, and Windows platforms

 On UNIX, Linux, and Windows platforms, sandboxing restricts which directories a Managed File Transfer Agent can read from and write to. When sandboxing is activated, the Managed File Transfer

Agent can read and write to the directories specified as allowed, and any subdirectories that the specified directories contain unless the subdirectories are specified as denied in the `sandboxRoot`. Managed File Transfer sandboxing does not take precedence over operating system security. The user that started the Managed File Transfer Agent must have the appropriate operating system level access to any directory to be able to read from or write to the directory. A symbolic link to a directory is not followed if the directory linked to is outside the specified `sandboxRoot` directories (and subdirectories).

Working in a sandbox on z/OS

z/OS On z/OS, sandboxing restricts the data set name qualifiers that the Managed File Transfer Agent can read from and write to. The user that started the Managed File Transfer Agent must have the correct operating system authorities to any data sets involved. If you enclose a `sandboxRoot` data set name qualifier value in double quotation marks, the value follows the normal z/OS convention and is treated as fully qualified. If you omit the double quotation marks, the `sandboxRoot` is prefixed with the current user ID. For example, if you set the `sandboxRoot` property to the following: `sandboxRoot="//test`, the agent can access the following data sets (in standard z/OS notation) `//username.test.**`. At run time, if the initial levels of the fully resolved data set name do not match the `sandboxRoot`, the transfer request is rejected.

Working in a sandbox on IBM i systems

IBM i For files in the integrated file system on IBM i systems, sandboxing restricts which directories a Managed File Transfer Agent can read from and write to. When sandboxing is activated, the Managed File Transfer Agent can read and write to the directories specified as allowed, and any subdirectories that the specified directories contain unless the subdirectories are specified as denied in the `sandboxRoot`. Managed File Transfer sandboxing does not take precedence over operating system security. The user that started the Managed File Transfer Agent must have the appropriate operating system level access to any directory to be able to read from or write to the directory. A symbolic link to a directory is not followed if the directory linked to is outside the specified `sandboxRoot` directories (and subdirectories).

Related reference

[“Additional checks for wildcard transfers” on page 49](#)

From IBM MQ 9.0.1 in Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 1 in Long Term Support, if an agent has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from, you can specify that additional checks are to be made on wildcard transfers for that agent.

[“Working with MFT agent sandboxes” on page 45](#)

To add an additional level of security to Managed File Transfer, you can restrict the area of a file system that an agent can access.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Working with MFT user sandboxes

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

User sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent.

To enable user sandboxing, add the following property to the `agent.properties` file for the agent that you want to restrict:

```
userSandboxes=true
```

When this property is present and set to true the agent uses the information in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/UserSandboxes.xml` file to determine which parts of the file system the user who requests the transfer can access.

The `UserSandboxes.xml` XML is composed of an `<agent>` element that contains zero or more `<sandbox>` elements. These elements describe which rules are applied to which users. The `user` attribute of the `<sandbox>` element is a pattern that is used to match against the MQMD user of the request.

The file `UserSandboxes.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

If you specify the `userPattern="regex"` attribute or value, the `user` attribute is interpreted as a Java regular expression. For more information, see [“Regular expressions used by MFT” on page 633](#).

If you do not specify the `userPattern="regex"` attribute or value the `user` attribute is interpreted as a pattern with the following wildcard characters:

- asterisk (*), which represents zero or more characters
- question mark (?), which represents exactly one character

Matches are performed in the order that the `<sandbox>` elements are listed in the file. Only the first match is used, all following potential matches in the file are ignored. If none of the `<sandbox>` elements specified in the file match the MQMD user associated with the transfer request message, the transfer cannot access the file system. When a match has been found between the MQMD user name and a `user` attribute, the match identifies a set of rules inside a `<sandbox>` element that are applied to the transfer. This set of rules is used to determine which files, or data sets, can be read from or written to as part of the transfer.

Each set of rules can specify a `<read>` element, which identifies which files can be read, and a `<write>` element which identifies which files can be written. If you omit the `<read>` or `<write>` elements from a set of rules, it is assumed that the user associated with that set of rules is not allowed to perform any reads or any writes, as appropriate.

Note: The `<read>` element must be before the `<write>` element, and the `<include>` element must be before the `<exclude>` element, in the `UserSandboxes.xml` file.

Each `<read>` or `<write>` element contains one or more patterns that are used to determine whether a file is in the sandbox and can be transferred. Specify these patterns by using the `<include>` and `<exclude>` elements. The `name` attribute of the `<include>` or `<exclude>` element specifies the pattern to be matched. An optional `type` attribute specifies whether the `name` value is a file or queue pattern. If the `type` attribute is not specified, the agent treats the pattern as a file or directory path pattern. For example:

```
<tns:read>
  <tns:include name="/home/user/**"/>
  <tns:include name="USER.**" type="queue"/>
  <tns:exclude name="/home/user/private/**"/>
</tns:read>
```

The `<include>` and `<exclude>` name patterns are used by the agent to determine whether files, data sets, or queues can be read from or written to. An operation is allowed if the canonical file path, data set, or queue name matches at least one of the included patterns and exactly zero of the excluded patterns. The patterns specified by using the `name` attribute of the `<include>` and `<exclude>` elements use the path separators and conventions appropriate to the platform that the agent is running on. If you specify relative file paths, the paths are resolved relative to the `transferRoot` property of the agent.

When specifying a queue restriction, a syntax of `QUEUE@QUEUEMANAGER` is supported, with the following rules:

- If the at character (@) is missing from the entry, the pattern is treated as a queue name that can be accessed on any queue manager. For example, if the pattern is `name` it is treated the same way as `name@**`.

- If the at character (@) is the first character in the entry, the pattern is treated as a queue manager name and all queues on the queue manager can be accessed. For example, if the pattern is @name it is treated the same way as **@name..

The following wildcard characters have special meaning when you specify them as part of the name attribute of the <include> and <exclude> elements:

A single asterisk matches zero or more characters in a directory name, or in a qualifier of a data set name or queue name.

?

A question mark matches exactly one character in a directory name, or in a qualifier of a data set name or queue name.

Two asterisk characters match zero or more directory names, or zero or more qualifiers in a data set name or queue name. Also, paths that end with a path separator have an implicit "***" added to the end of the path. So /home/user/ is the same as /home/user/**.

For example:

- /**/test/** matches any file that has a test directory in its path
- /test/file? matches any file inside the /test directory that starts with the string file followed by any single character
- c:\test*.txt matches any file inside the c:\test directory with a .txt extension
- c:\test***.txt matches any file inside the 'c:\test directory, or one of its subdirectories that has a .txt extension
-  // 'TEST.*.DATA' matches any data set that has the first qualifier of TEST, has any second qualifier, and a third qualifier of DATA.
- *@QM1 matches any queue on the queue manager QM1 that has a single qualifier.
- TEST.*.QUEUE@QM1 matches any queue on the queue manager QM1 that has the first qualifier of TEST, has any second qualifier, and a third qualifier of QUEUE.
- **@QM1 matches any queue on the queue manager QM1.

Symbolic links

You must fully resolve any symbolic links that you use in file paths in the UserSandboxes.xml file by specifying hard links in the <include> and <exclude> elements. For example, if you have a symbolic link where /var maps to /SYSTEM/var, you must specify this path as <tns:include name="/SYSTEM/var"/>, otherwise the intended transfer fails with a user sandbox security error.

Example

This example shows how to allow the user with the MQMD user name guest to transfer any file from the /home/user/public directory or any of its subdirectories on the system where the agent AGENT_JUPITER is running, by adding the following <sandbox> element to the file UserSandboxes.xml in AGENT_JUPITER's configuration directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
  <tns:agent>
    <tns:sandbox user="guest">
      <tns:read>
        <tns:include name="/home/user/public/**"/>
      </tns:read>
    </tns:sandbox>
  </tns:agent>
</tns:userSandboxes>
```

```
</tns:agent>
</tns:userSandboxes>
```

Example

This example shows how to allow any user with the MQMD user name account followed by a single digit, for example account4, to complete the following actions:

- Transfer any file from the /home/account directory or any of its subdirectories, excluding the /home/account/private directory on the system where the agent AGENT_SATURN is running
- Transfer any file to the /home/account/output directory or any of its subdirectories on the system where the agent AGENT_SATURN is running
- Read messages from queues on the local queue manager starting with the prefix ACCOUNT . unless it starts with ACCOUNT .PRIVATE . (that is has PRIVATE at the second level).
- Transfer data onto queues starting with the prefix ACCOUNT .OUTPUT . on any queue manager.

To allow a user with the MQMD user name account to complete these actions, add the following <sandbox> element to the file UserSandboxes.xml, in AGENT_SATURN's configuration directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
  <tns:agent>
    <tns:sandbox user="account[0-9]" userPattern="regex">
      <tns:read>
        <tns:include name="/home/account/**"/>
        <tns:include name="ACCOUNT.**" type="queue"/>
        <tns:exclude name="ACCOUNT.PRIVATE.**" type="queue"/>
        <tns:exclude name="/home/account/private/**"/>
      </tns:read>
      <tns:write>
        <tns:include name="/home/account/output/**"/>
        <tns:include name="ACCOUNT.OUTPUT.**" type="queue"/>
      </tns:write>
    </tns:sandbox>
  </tns:agent>
</tns:userSandboxes>
```

Related reference

[“Additional checks for wildcard transfers” on page 49](#)

From IBM MQ 9.0.1 in Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 1 in Long Term Support, if an agent has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from, you can specify that additional checks are to be made on wildcard transfers for that agent.

[“Working with MFT user sandboxes” on page 46](#)

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, agent.properties, that must contain the information that an agent uses to connect to its queue manager. The agent.properties file can also contain properties that alter the behavior of the agent.

Additional checks for wildcard transfers

From IBM MQ 9.0.1 in Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 1 in Long Term Support, if an agent has been configured with a user or agent sandbox in order to restrict the locations that the agent

can transfer files to and from, you can specify that additional checks are to be made on wildcard transfers for that agent.

additionalWildcardSandboxChecking property

To enable additional checking for wildcard transfers, add the following property to the `agent.properties` file for the agent that you want to check.

```
additionalWildcardSandboxChecking=true
```

When this property is set to true, and the agent makes a transfer request that attempts to read a location that is outside the defined sandbox for file matching of the wildcard, the transfer fails. If there are multiple transfers within one transfer request, and one of these requests fails due to it attempting to read a location outside of the sandbox, the entire transfer fails. If checking fails, the reason for failure is given in an error message.

If the `additionalWildcardSandboxChecking` property is omitted from an agent's `agent.properties` file or is set to false, no additional checks are made on wildcard transfers for that agent.

Error messages for wildcard checking

From IBM MQ 9.0.1 in Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 1 in Long Term Support, the messages that are reported when a wildcard transfer request is made to a location outside a configured sandbox location have changed.

The following message occurs when a wildcard file path in a transfer request is located outside of the restricted sandbox:

```
BFGSS0077E: Attempt to read file path: path has been denied.  
The file path is located outside of the restricted transfer sandbox.
```

The following message occurs when a transfer within a multiple transfer request contains a wildcard transfer request where the path is located outside of the restricted sandbox:

```
BFGSS0078E: Attempt to read file path: path has been ignored as another transfer  
item in the managed transfer attempted to read outside of the restricted transfer sandbox.
```

The following message occurs when a file is located outside of the restricted sandbox:

```
BFGSS0079E: Attempt to read file file path has been denied.  
The file is located outside of the restricted transfer sandbox.
```

The following message occurs in a multiple transfer request where another wildcard transfer request has caused this one to be ignored:

```
BFGSS0080E: Attempt to read file: file path has been ignored as another transfer  
item in the managed transfer attempted to read outside of the restricted transfer sandbox.
```

In the case of single file transfers that do not include wildcards, the message that is reported when the transfer involves a file that is located outside of the sandbox is unchanged from earlier releases:

```
Fails with BFGI00056E: Attempt to read file "FILE" has been denied.  
The file is located outside of the restricted transfer sandbox.
```

Related reference

[“Working with MFT user sandboxes” on page 46](#)

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

[“Working with MFT agent sandboxes” on page 45](#)

To add an additional level of security to Managed File Transfer, you can restrict the area of a file system that an agent can access.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Configuring SSL or TLS encryption for MFT

You can use SSL or TLS can be used with IBM MQ Managed File Transfer to secure the communication between agents and their agent queue managers, commands and the queue managers that they are connecting to, and the various queue manager to queue manager connections within your topology.

Before you begin

You can use SSL or TLS encryption to encrypt messages that are flowing through an IBM MQ Managed File Transfer topology. These include:

- Messages that pass between an agent and its agent queue manager.
- Messages for commands and the queue managers that they are connecting to.
- Internal messages that flow between the agent queue managers, command queue managers and coordination queue manager within the topology.

About this task

For general information about using SSL with IBM MQ, see [Working with SSL or TLS](#). In IBM MQ terms, Managed File Transfer is a standard Java client application.

Follow these steps to use SSL with Managed File Transfer:

Procedure

1. Create a truststore file and optionally a keystore file (these files can be the same file). If you do not need client-authentication (that is, `SSLCAUTH=OPTIONAL` on channels) you do not need to provide a keystore. You require a truststore only to authenticate the queue manager's certificate.

The key algorithm used for creating certificates for the truststore and keystores must be RSA to work with IBM MQ.

2. Set up your IBM MQ queue manager to use SSL.

For information about setting up a queue manager to use SSL using IBM MQ Explorer for example, see [Configuring SSL on queue managers](#).

3. Save the truststore file and keystore file (if you have one) in a suitable location. A suggested location is the `config_directory/coordination_qmgr/agents/agent_name` directory.
4. Set the SSL properties as required for each SSL-enabled queue manager in the appropriate Managed File Transfer properties file. Each set of properties refers to a separate queue manager (agent, coordination, and command), although one queue manager might perform two or more of these roles.

One of the **CipherSpec** or **CipherSuite** properties is required, otherwise the client tries to connect without SSL. Both the **CipherSpec** or **CipherSuite** properties are provided because of the terminology differences between IBM MQ and Java. Managed File Transfer accepts either property and does the necessary conversion, so you do not need to set both properties. If you do specify both the **CipherSpec** or **CipherSuite** properties, **CipherSpec** takes precedence.

The **PeerName** property is optional. You can set the property to the Distinguished Name of the queue manager that you want to connect to. Managed File Transfer rejects connections to an incorrect SSL server with a Distinguished Name that does not match.

Set the **SslTrustStore** and **SslKeyStore** properties to file names that point to the truststore and keystore files. If you are setting up these properties for an agent that is already running, stop and restart the agent to reconnect in SSL mode.

Properties files contain plain-text passwords so consider setting appropriate file system permissions.

For more information about SSL properties, see [“SSL properties for MFT” on page 580](#).

5. If an agent queue manager uses SSL, you cannot provide the necessary details when you create the agent. Use the following steps to create the agent:
 - a) Create the agent by using the **fteCreateAgent** command. You receive a warning about being unable to publish the existence of the agent to the coordination queue manager.
 - b) Edit the `agent.properties` file that was created by the previous step to add the SSL information. When the agent is successfully started, the publish is attempted again.
6. If agents or instances of the IBM MQ Explorer are running while the SSL properties in the `agent.properties` file or `coordination.properties` file are changed, you must restart the agent or IBM MQ Explorer.

Related reference

“The MFT `agent.properties` file” on page 526

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Connecting to a queue manager in client mode with channel authentication

IBM WebSphere MQ 7.1 introduced channel authentication records to control more precisely access at a channel level. This change in behavior means that by default newly created IBM WebSphere MQ 7.1 or later queue managers reject client connections from the Managed File Transfer component.

For more information about channel authentication, see [Channel authentication records](#).

If the channel authentication configuration for the SVRCONN used by Managed File Transfer specifies a non-privileged MCAUSER ID, you must grant specific authority records for the queue manager, queues, and topics, to allow the Managed File Transfer Agent and commands to work correctly. Use the MQSC command [SET CHLAUTH](#) or the PCF command [Set Channel Authentication Record](#) to create, modify, or remove channel authentication records. For all Managed File Transfer agents that you want to connect to the IBM WebSphere MQ 7.1 or later queue manager, you can either set up an MCAUSER ID to use for all your agents, or set up a separate MCAUSER ID for each agent.

Grant each MCAUSER ID the following permissions:

- Authority records required for the queue manager:
 - connect
 - setid
 - inq
- Authority records required for queues.

For all agent-specific queues, that is queue names that end in `agent_name` in the following list, you must create these queue authority records for each agent that you want to connect to the IBM WebSphere MQ 7.1 or later queue manager by using a client connection.

- put, get, dsp (SYSTEM.DEFAULT.MODEL.QUEUE)
- put, get, setid, browse (SYSTEM.FTE.COMMAND.*agent_name*)
- put, get (SYSTEM.FTE.DATA.*agent_name*)
- put, get (SYSTEM.FTE.REPLY.*agent_name*)
- put, get, inq, browse (SYSTEM.FTE.STATE.*agent_name*)
- put, get, browse (SYSTEM.FTE.EVENT.*agent_name*)
- put, get (SYSTEM.FTE)
- Authority records required for topics:
 - sub, pub (SYSTEM.FTE)
- Authority records required for file transfers.

If you have separate MCAUSER IDs for source and destination agent, create the authority records on agents' queues at both source and destination.

For example, if the source agent's MCAUSER ID is **user1** and the destination agent MCAUSER ID is **user2**, set the following authorities for the agent users:

AGENT user	Queue	Authority required
user1	SYSTEM.FTE.DATA.destination_agent_name	put
user1	SYSTEM.FTE.COMMAND.destination_agent_name	put
user2	SYSTEM.FTE.REPLY.source_agent_name	put
user2	SYSTEM.FTE.COMMAND.source_agent_name	put

Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node

Configure the Connect:Direct bridge agent and the Connect:Direct node to connect to each other through the SSL protocol by creating a keystore and a truststore, and by setting properties in the Connect:Direct bridge agent properties file.

About this task

These steps include instructions for getting your keys signed by a certificate authority. If you do not use a certificate authority, you can generate a self-signed certificate. For more information about generating a self-signed certificate, see [Working with SSL or TLS on UNIX and Windows systems](#).

These steps include instructions for creating a new keystore and truststore for the Connect:Direct bridge agent. If the Connect:Direct bridge agent already has a keystore and truststore that it uses to connect securely to IBM MQ queue managers, you can use the existing keystore and truststore when connecting securely to the Connect:Direct node. For more information, see [“Configuring SSL or TLS encryption for MFT” on page 51](#).

Procedure

For the Connect:Direct node, complete the following steps:

1. Generate a key and signed certificate for the Connect:Direct node.
You can do this by using the IBM Key Management tool that is provided with IBM MQ. For more information, see [Working with SSL or TLS](#).
2. Send a request to a certificate authority to have the key signed. You receive a certificate in return.
3. Create a text file; for example, `/test/ssl/certs/CACert`, that contains the public key of your certification authority.
4. Install the Secure+ Option on the Connect:Direct node.
If the node already exists, you can install the Secure+ Option by running the installer again, specifying the location of the existing installation, and choosing to install only the Secure+ Option.
5. Create a new text file; for example, `/test/ssl/cd/keyCertFile/node_name.txt`.
6. Copy the certificate that you received from your certification authority and the private key, located in `/test/ssl/cd/privateKeys/node_name.key`, into the text file.

The contents of `/test/ssl/cd/keyCertFile/node_name.txt` must be in the following format:

```
-----BEGIN CERTIFICATE-----
MIICnzCCAigAwIBAgIBGjANBgkqhkiG9w0BAQUFADBBeMQswCQYDVQQGEwJHqjES
MBAGA1UECBMJSgFtchNoaXJlMRAdBgYDVQQHEwdIdXJzbGV5MQwwCgYDVQQKEwNJ
Qk0xDjAMBgNVBAwTBU1RSVBUMQswCQYDVQQDEwJDQTAeFw0xMTAzMDEwNDZa
Fw0yMTAyMjYxNjIwNDZaMFExCzAJBgNVBAYTAkdCMRlWEAyDVQQIEwLIYW1wc2hp
cmUxDDAKBgNVBAoTA0lCTTEOMAwGA1UECzMFTVFGEUxZDzANBgNVBAMTBmJpbmJh
ZzCBnzANBgkqhkiG9w0BAQEFAA0bjQAwwYkCgYEAvgP1QIk1U9ypSKD1Xo0Do1yk
```

```

EyMFXB0UpZRrDVxjoSEC0vtWncJ199e+Vc4UpNybdyBu+Nkd1MNoF4QxeQcLAFj
WnhakqCiQ+JIAD5AurhnriwChe0MV3kjA84GKH/r0SVqt1984mu/1DyS819XcfSSn
c00MsK1KbneVSCIV2XECaWEAAa7MHkwCQYDVR0TBAlwADAsBg1ghkgBhvCAQ0E
HxYdT3B1b1NTTCBHZW51cmF0ZWQgQ2VydG1maWNoGUmwHQYDVR00BBYEFNXMIpSc
csBXUniW4A3UzrZnCRsv3MB8GA1UdIwQYMBaAFDXY8rmj41Vz5+FVAoQb++cns+B4
MA0GCSqGSIb3DQEBBQUAA4GBAFc7k1Xa4pGKYgwxKpE3ZF6FNwy4vBXS216/ja
8h/v18+iv010CL8t0ZOKSU95fyZLzOPKnCH7v+ItFSE3CIIEk9D1z2U6W091ICwn
17PL72Tdfal3kabwHYVf17IVcuL+VZsZ3HjLggP2qH09ZuJPspeT9+AxFVMLiaAb
8eHw

```

```

-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,64A02DA15B6B6EF9

```

```

57kqxL0J/gRU0IQ6hVK2YN13B4E1jAi1gSme0I5ZpEIG8CHXISKB7/0cke2FTqsV
lvI99QyCxsDwoMnt5fj51v7aPmVeS60b0m+U1Gre8B/Ze18JvJ204K2U72rDCXE
5e6eFxDuM207sQDy20euBVELJtM2k0kL1R0doQqS1U3XQNgJw/t3ZIx5hPXWEQT
rjRQ064BEhb+PzzxPF8uwzZ9IrkUK9BJ/UUnqC60dBR87IeA4pnJD1Jvb2ML7EN9Z
5Y+50hTKI80GvBvWX04fHyvIX5aslwhBoArXIS1AtNtRptPvoaP1zyIAeZ60Cvo/
Sfo+A2UhmTEJe0JaZG2XZ3H495fAw/EHmjehzIACwukQ9nSIETgu4A1+CV64RJED
aYBCM8UjaAkBZDH5gn7+eBov0ssXAXWdyJBVhU0jXjvAj/e1h+kcSF1hax5D//AI
66nRMZzboSxNqkjcVd8wfdwP+bEjDzUaaaarJTS71IFeLLw7eJ8MNAkMGicDkycL0
EPBU9X5QnHKLKOfYHN/1WgUk8qt3UytFXXfzTXGF3EbsWbBupkT5e5+1YcX80VZ6
sHFPN1HluCny/riUcBy9iviVeodX8Iom0chSy05DK18bwZNjYtUP+CtYHNFU5BaD
I+1uU0AeJ+wjQYKT1WaeIGZ3VxuNITJu18y5qDTXXfX7vxM50oWxa6U5+AYuGUMg
/itPZmUmNrhjT7ghT6i1IQ0aBowXXKJB1Mmq/6BQXN2IhkD9ys2qrvM1hdi5nAf
egmdiG50l0LnBRqWbFR+DykpAhK4SaDi2F52Uxovw3Lh1w8dQP71zQ==
-----END RSA PRIVATE KEY-----

```

7. Start the Secure+ Admin Tool.

- On Linux or UNIX systems, run the command **spadmin.sh**.
- On Windows systems, click **Start > Programs > Sterling Commerce Connect:Direct > CD Secure+ Admin Tool**

The CD Secure+ Admin Tool starts.

8. In the CD Secure+ Admin Tool, double-click the **.Local** line to edit the main SSL or TLS settings.

- Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.
- Select **Disable Override**.
- Select at least one Cipher Suite.
- If you want two-way authentication, change the value of **Enable Client Authentication** to Yes.
- In the **Trusted Root Certificate** field, enter the path to the public certificate file of your certification authority, `/test/ssl/certs/CAcert`.
- In the **Key Certificate File** field, enter the path to the file that you created, `/test/ssl/cd/keyCertFile/node_name.txt`.

9. Double-click the **.Client** line to edit the main SSL or TLS settings.

- Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.
- Select **Disable Override**.

For the Connect:Direct bridge agent, perform the following steps:

10. Create a truststore. You can do this by creating a dummy key and then deleting the dummy key.

You can use the following commands:

```
keytool -genkey -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
```

```
keytool -delete -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
```

11. Import the public certificate of the certification authority into the truststore.

You can use the following command:

```
keytool -import -trustcacerts -alias myCA
```

```
-file /test/ssl/certs/CAcert  
-keystore /test/ssl/fte/stores/truststore.jks
```

12. Edit the Connect:Direct bridge agent properties file.

Include the following lines anywhere in the file:

```
cdNodeProtocol=protocol  
cdNodeTruststore=/test/ssl/fte/stores/truststore.jks  
cdNodeTruststorePassword=password
```

In the example in this step, *protocol* is the protocol you are using, either SSL or TLS, and *password* is the password that you specified when you created the truststore.

13. If you want two-way authentication, create a key and certificate for the Connect:Direct bridge agent.

a) Create a keystore and key.

You can use the following command:

```
keytool -genkey -keyalg RSA -alias agent_name  
-keystore /test/ssl/fte/stores/keystore.jks  
-storepass password -validity 365
```

b) Generate a signing request.

You can use the following command:

```
keytool -certreq -v -alias agent_name  
-keystore /test/ssl/fte/stores/keystore.jks -storepass password  
-file /test/ssl/fte/requests/agent_name.request
```

c) Import the certificate you receive from the preceding step into the keystore. The certificate must be in x.509 format.

You can use the following command:

```
keytool -import -keystore /test/ssl/fte/stores/keystore.jks  
-storepass password -file certificate_file_path
```

d) Edit the Connect:Direct bridge agent properties file.

Include the following lines anywhere in the file:

```
cdNodeKeystore=/test/ssl/fte/stores/keystore.jks  
cdNodeKeystorePassword=password
```

In the example in this step, *password* is the password that you specified when you created the keystore.

Related tasks

[“Configuring the Connect:Direct bridge” on page 135](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Configuring MFT

You can configure the features of Managed File Transfer after installation.

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

[“MFT recovery and restart” on page 235](#)

If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, Managed File Transfer recovers as follows in these scenarios:

[“Hints and tips for using MFT” on page 261](#)

Here are some suggestions to help you to make best use of Managed File Transfer.

[“Administering MFT” on page 146](#)

Use Managed File Transfer commands to administer Managed File Transfer. You can also use the IBM MQ Explorer for some of the administrative tasks.

Related tasks

[“Monitoring MFT resources” on page 162](#)

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

[“Configuring MQMFTCredentials.xml on multiplatforms” on page 40](#)

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT commands that connect with a queue manager to supply user ID and password credentials. Similarly, MFT loggers might be required to specify a user ID and password when connecting to a database. This credential information can be stored in the MFT credentials file.

Related reference

[“MFT commands” on page 356](#)

All Managed File Transfer commands are listed with links to their detailed descriptions.

[“Securing MFT” on page 38](#)

Directly after installation and with no modification, Managed File Transfer has a level of security that might be suitable for test or evaluation purposes in a protected environment. However, in a production environment, you must consider appropriately controlling who can start file transfer operations, who can read and write the files being transferred, and how to protect the integrity of files.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Troubleshooting MFT” on page 259](#)

Use the following reference information to help you to diagnose errors in Managed File Transfer:

MFT configuration options on Multiplatforms

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

You can have multiple sets of configuration options, each set of configuration options contains a set of directories and properties files. The values defined in these properties files are used as the default parameters for all Managed File Transfer commands, unless you explicitly specify a different value on the command line.

To change the default set of configuration options that you are using you can use the **fteChangeDefaultConfigurationOptions** command. To change the set of configuration options that you are using for an individual command you can use the **-p** parameter with any Managed File Transfer command.

The name of a set of configuration options is the name of the coordination queue manager, and it is recommended that this is not changed. However, it is possible to change the name of a set of

configuration options but you must change the name of the `config` and `logs` directories. In the following examples, the name of the set of configuration options is represented as `coordination_qmgr_name`.

Configuration options directory structure

When you configure the product, directories and properties files are created in the following structure in the configuration directory. You can also change these directories and properties files with the following commands: **fteSetupCoordination**, **fteSetupCommands**, **fteChangeDefaultConfiguration**, and **fteCreateAgent**.

```
MQ_DATA_PATH/mqft/  
  config/  
    coordination_qmgr_name/  
      coordination.properties  
      command.properties  
      agents/  
        agent_name/  
          agent.properties  
          exits  
        loggers/  
          logger_name  
            logger.properties  
      installations/  
        installation_name/  
          installation.properties
```

The `coordination_qmgr_name` directory is a configuration options directory. There can be more than one configuration options directory in the configuration directory. The `agent_name` directory is an agent directory. In addition to containing the `agent.properties` file, this directory contains the `exits` directory, which is the default location for user exit routines and various XML files generated by the **fteCreateBridgeAgent** and **fteCreateCDAgent** commands. There can be more than one agent directory in the `agents` directory of a set of configuration options.

Properties files

installation.properties

The `installation.properties` file specifies the name of your default set of configuration options. This entry points Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager. For more information about the `installation.properties` file, see [“The MFT installation.properties file” on page 513](#).

coordination.properties

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive. For more information about the `coordination.properties` file, see [“The MFT coordination.properties file” on page 517](#).

command.properties

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager. For more information about the `command.properties` file, see [“The MFT command.properties file” on page 522](#).

agent.properties

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent. For more information about the `agent.properties` file, see [“The MFT agent.properties file” on page 526](#).

logger.properties

The `logger.properties` file specifies the configuration properties for the loggers. For more information about the `logger.properties` file, see [“MFT logger configuration properties” on page 111](#).

Properties files and code pages

The content of all the Managed File Transfer properties files must remain in US English because of a limitation of Java. If you edit properties files on a non-US English system, you must use Unicode escape sequences.

z/OS MFT configuration options on z/OS

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

For more information about configuration options on [Multiplatforms](#), see [“MFT configuration options on Multiplatforms” on page 56](#).

On z/OS, the configuration location is defined by the environment variable `BFG_DATA`. If a configuration does not already exist under the UNIX System Services directory that is referenced by `BFG_DATA`, the `BFGCUSTOM` JCL script of an MFT command PDSE library data set generates the jobs required to create the configuration. The configuration is then created when you run these generated jobs. Configuration creation relies on `BFG_DATA` referencing an existing directory that is accessible.

You can also create and maintain a configuration by using the same **fte** commands that are available on both Multiplatforms and z/OS. For a list of the **fte** commands, see [“MFT commands” on page 356](#).

V 9.0.1 Configuring the Redistributable Managed File Transfer Agent

Configure the optionally provided Redistributable Managed File Transfer Agent to connect to an existing IBM MQ infrastructure and enable users to transfer files without installing IBM MQ to get the Managed File Transfer functionality.

Before you begin

Redistributable Managed File Transfer Agent provides the functionality of Managed File Transfer with these exceptions:

- Bindings mode connection to the coordination, command, and agent queue managers, is not supported, you must use client mode connection. When you are issuing commands, you must provide the parameters that are optional when you are using the Managed File Transfer that is installed as part of IBM MQ: queue manager host, port, name, and channel name.
- The following commands are not included:
 - `fteCreateCDAgent.cmd`
 - `fteCreateLogger.cmd`
 - `fteDeleteLogger.cmd`
 - `fteMigrateLogger.cmd`
 - `fteSetLoggerTraceLevel.cmd`
 - `fteShowLoggerDetails.cmd`
 - `fteStartLogger.cmd`
 - `fteStopLogger.cmd`

For a complete list of available commands, see [Installed MFT command sets](#).

- Managed File Transfer Connect:Direct is not supported.
- IBM MQ Explorer is not included.

Windows You must install the following Microsoft libraries on your system to use Redistributable Managed File Transfer Agent:

- Microsoft Visual C++ Redistributable 2008
- Microsoft Visual C++ Redistributable 2012

These libraries are available from Microsoft. See [The latest supported Visual C++ downloads](#).

About this task

You want to allow users to transfer files between their local environment and the existing IBM MQ infrastructure without the need for them to install IBM MQ to be able to do so. Follow these steps to download and extract the Redistributable Managed File Transfer Agent:

Procedure

1. Download the Redistributable Managed File Transfer Agent package from [IBM Fix Central](#).

a) Choose the package for your operating system:

- **Windows** 9.0.1.0-IBM-MQFA-Redist-Win64
- **Linux** 9.0.1.0-IBM-MQFA-Redist-LinuxX64
- **Linux** 9.0.1.0-IBM-MQFA-Redist-LinuxS390X
- **Linux** 9.0.1.0-IBM-MQFA-Redist-LinuxPPC64LE

b) Identify the directory where you want to extract the package, for example:

- **Windows** C:\MFTZ
- **Linux** /home/MFTZ

2. Extract the contents of the downloaded package:

- **Windows** Use the Windows Explorer tools to extract.
- **Linux** Extract and untar as follows:

```
gunzip 9.0.1.0-IBM-MQFA-Redist-LinuxX64.tar.gz
```

and then

```
tar xvf 9.0.1.0-IBM-MQFA-Redist-LinuxX64.tar
```

The following directories are created:

- **Windows** **Linux** bin: Contains all the required MFT commands
- **Windows** bin64: Contains required libraries that are needed for Windows 64-bit OS support
- **Windows** **Linux** java: Contains the IBM JRE and IBM MQ libraries
- **Windows** **Linux** licenses: Contains the license files
- **Windows** **Linux** mqft: Contains ant and lib directories that are required for Ant support and for core MFT function support
- **Windows** **Linux** swtag: Contains swidtag file that is required by license managers to identify the installations on the machine

What to do next

You are ready to configure the MFT Agent. For the next steps, see the following topic: [“Creating the initial configuration” on page 60.](#)

Related reference

[“Possible errors when configuring the Redistributable MFT Agent” on page 306](#)

Error messages when you are configuring the Redistributable Managed File Transfer Agent

Related information

[Video: How to install and configure the Redistributable Managed File Transfer Agent \(YouTube\)](#)

Creating the initial configuration

You can configure a Managed File Transfer Agent to connect to an existing IBM MQ configuration.

Before you begin

V 9.0.1 Make sure that you download and extract the contents of the Redistributable Managed File Transfer Agent package. For more information, see [“Configuring the Redistributable Managed File Transfer Agent” on page 58.](#)

About this task

Create the environment to be able to use commands for setting up the connectivity with the queue manager that is running on the IBM MQ server.

Procedure

1. Create the environment for the Redistributable Managed File Transfer Agent.

When you run the **fteCreateEnvironment** command, the MFT data directory with the configuration information for MFT Agents is created. Ensure that you are in the `bin` directory that was created when you extracted the downloaded MFT Agent redistributable component. Run the following command:

- **Windows**

```
fteCreateEnvironment.cmd -d datapath location
```

- **Linux**

```
./fteCreateEnvironment -d datapath location
```

This command takes one optional parameter:

-d

Location for data path where the MFT configuration is created, stored, and maintained. If the user runs the **fteCreateEnvironment** without specifying the data location, the directory `mftdata` is created in the location where the Redistributable Managed File Transfer Agent is extracted.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

You can also set the environment variable `BFG_DATA` with the data path location:

```
BFG_DATA=Datapath location
```

Before creating, starting, and stopping an agent, or any other commands, you must ensure that the `BFG_DATA` variable is set to the correct data path location.

2. Set up the IBM MQ connectivity.
 - a) Set up the coordination queue manager by using the **fteSetupCoordination** command.

The **fteSetupCoordination** command creates the set that is up required for coordination queue managers and the directories that are needed for further configuration. Redistributable Managed File Transfer Agent works in client mode so you must provide extra parameters with this command to avoid an error, as bindings mode is not supported.

```
fteSetupCoordination -coordinationQMgr PRMFTDEM02
                    -coordinationQMgrHost 9.121.59.233 -coordinationQMgrPort 3002
                    -coordinationQMgrChannel SYSTEM.DEF.SVRCONN
```

For more details and steps for using the **fteSetupCoordination** command, see [“fteSetupCoordination” on page 492](#). For information on how to configure the coordination queue manager, see [“Configuring the coordination queue manager for MFT” on page 92](#).

b) Create and set up the command queue manager:

```
fteSetupCommands -p PRMFTDEM02 -connectionQMgrHost 9.121.59.233
                 -connectionQMgrPort 3002 -connectionQMgrChannel SYSTEM.DEF.SVRCONN
                 -connectionQMgr PRMFTDEM02 -f
```

For more details and steps for using the **fteSetupCommands** command, see [“fteSetupCommands: create the MFT command.properties file” on page 490](#).

3. Create MFT Agent definition for an end point.

```
fteCreateAgent -p PRMFTDEM02 -agentQMgrHost 9.121.59.233
               -agentQMgrPort 3002 -agentQMgrChannel SYSTEM.DEF.SVRCONN
               -agentName AGENT.TRI.BANK -agentQMgr PRMFTDEM02 -f
```

For more information on using the **fteCreateAgent** command to configure an agent and the agent queue manager, see [“fteCreateAgent \(create an MFT agent\)” on page 371](#).

In steps [“2” on page 60](#) and [“3” on page 61](#) for each agent, you create queue and topic definitions on the agent queue manager.

4. Start the agent and you are ready to transfer files.

```
fteStartAgent -p PRMFTDEM02 AGENT.TRI.BANK
```

You can verify the status of the agent by running the following command:

```
ftelistAgents
```

For more details on using the **ftelistAgents** command, see [“ftelistAgents: list the MFT agents for a coordination queue manager” on page 452](#).

Related concepts

[“Configuring MFT” on page 55](#)

You can configure the features of Managed File Transfer after installation.

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Creating an MFT Agent or Logger command data set

You can create a PDSE data set of commands from the Managed File Transfer command template data set for a specific Managed File Transfer Agent or Managed File Transfer Logger for a specific coordination.

About this task

Complete the following steps:

Procedure

1. Make a copy of the MFT command template PDSE library data set SBFGCMD5.

 SBFGCMD5 must be copied into a new library, for example *prefix.agent.JCL_*. You can use an updated version of the SBFGCMD5(BFGCOPY) member with the following replacements:

- Replace *++supplied-library++* with the fully qualified name of the SBFGCMD5 PDSE.
 -  Replace *++service-library++* with the fully qualified name of the new MFT command PDSE library data set. The *++service-library++* is the output data set for the agent or logger service that is created.
2. For the new MFT command PDSE library data set, edit the member BFGCUSTM, which is a JCL script to customize the commands for the agent or logger. Each variable is specified in the format: *++variable name++*, which you must replace with its required value. For a description of the various JCL variables, see “z/OS JCL variables” on page 73. The BFGSTDIN DD statement defines variables in three categories: Variables, Properties, and Environment. The statement has the following format:

```
[Variables]
variable1=value1
variable2=value2
...
variableN=valueN
[Properties]
property1=property value1
property2=property value2
...
propertyN=property valueN
[Environment]
custom_variable1=value1
custom_variable2=value2
...
custom_variableN=valueN
```

Variables define the set of setup and environment variables that are required for each command.

Properties define overrides for the MFT configuration properties. You can add agent and logger properties as required to customize the agent or logger for your environment. For a list of all properties, see “[Configuration properties files](#)” on page 85. This facility is provided to save having to access the MFT configuration properties files, which are maintained as UNIX System Services files.

Environment defines any additionally required custom environment variables.

3. Submit job BFGCUSTM for the new MFT command PDSE library data set. This job generates the set of JCL commands, as new members of the PDSE, appropriate for the agent or logger. For a full list of the commands, see “[z/OS agent and logger command JCL scripts](#)” on page 78.

Job BFGCUSTM updates the library containing the JCL which includes a DD statement with DISP=OLD. You must exit the editor after submission to allow the job to execute.

Examine the output job log to check that the JCL script ran successfully. If there are any failures, correct them and submit the BFGCUSTM job again.

The BFGCUSTM JCL script also updates the UNIX System Services MFT configuration properties files as necessary to keep the files in step. If the configuration defined by the CoordinationQMgr property does not exist, warning messages are output and you must run the generated BFGCFGR and BFGCMCR jobs to create the configuration properties files. You must run BFGAGCR for an agent, and BFGLGCRS for a logger edit. If the specified configuration already exists, the configuration is updated with any properties as defined in the BFTCUSTM JCL script.

Related concepts

[“MFT configuration options on z/OS” on page 58](#)

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

Related tasks

[“Updating an existing MFT Agent or Logger command data set on z/OS” on page 73](#)

You can update a Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

z/OS **Configuring Managed File Transfer for z/OS**

Managed File Transfer for z/OS requires customization to enable the component to operate correctly.

About this task

You need to:

1. Edit a PDSE member to specify configuration data
2. Define the coordination queue manager.
3. Define the command queue manager
4. Configure one or more agents
5. Optionally: configure a logger task to store data in Db2

The sequence of tasks you need to perform is detailed in the following topics.

Related concepts

[“Reviewing the MFT configuration” on page 63](#)

You need to review the configuration of your system before you begin.

Related tasks

[Installing Managed File Transfer for z/OS](#)

z/OS **Reviewing the MFT configuration**

You need to review the configuration of your system before you begin.

Managed File Transfer (MFT) requires one or more queue managers to act in the following roles for each defined MFT configuration:

- A coordination queue manager, which maintains information on the status of each agent in the configuration published to a topic on the coordinator.
- One or more command or connection queue managers that act as the entry point to the IBM MQ network for MFT commands.
- One or more agent queue managers that provide the communication between an MFT agent and the IBM MQ network.

Each of the above roles can be performed by a separate queue manager, or you can combine the roles, so that, in the simplest configuration, all roles are performed by a single queue manager.

If you are adding a z/OS queue manager to an existing MFT environment you need to define connectivity between the z/OS queue manager and the other queue managers in the configuration. You can achieve this with manually defined transmission queues, or by the use of clustering.

Each MFT agent communicates with a single queue manager. If multiple agents communicate with the same queue manager, then the agent queue manager will have multiple queues defined for each agent:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHADM1.*agent_name*

Note that you can define generic security profiles, where you use a profile such as SYSTEM.FTE.COMMAND.* , or you can define specific profiles for each agent.

Related concepts

“Before you start” on page 64

Managed File Transfer (MFT) configuration uses files in UNIX System Services (USS) and PDSE data sets.

Before you start

Managed File Transfer (MFT) configuration uses files in UNIX System Services (USS) and PDSE data sets.

Most of the configuration and operation is done using JCL from a PDSE and you need to be familiar with working in an USS environment.

You can access OMVS from ISPF or you can use a Telnet type session using commands on your workstation, for example, Telnet Putty or SSH.

If you use OMVS from ISPF you can use the standard ISPF editor and browse commands **oedit** and **obrowse**.

You need to be familiar with the following USS commands

Command	Function
ls -ltr path	Lists information about the files in path
ls -ltrd directory	Lists information about the specified directory rather than the files in the directory.
find path -name xxx	Search for file named xxx in the path directory. xxx is case sensitive and can be like *zzz
chmod xxx path	Change files access permissions
df -k path	Reports how much free space remains in the files system. -k reports the free space in KB.
du -kt path	Reports the sizes of directories under path. Size reported in KB
oedit filename	Edit a file in OMVS
obrowse filename	Browse the file name

Review the items in the following table and complete the table with the appropriate entries for your enterprise. You need these values when you edit member `BFGCUSTM`.

<i>Table 6. Parameters needed for member BFGCUSTM</i>		
Name	Example data	Comments
BFG_JAVA_HOME	/java/java71_bit64_GA/ J7.1_64/	
BFG_GROUP_NAME	MQM	
LIBRARY	SCEN.FTE.JCL	Name of the MFT PDSE. You need a copy for each agent or logger task.
TMPDIR	/tmp	Read and write accessible USS path for temporary files.
SERVICE_TYPE	AGENT or LOGGER	
NAME	AGENT1	
BFG_PROD	/var/ibm/wmqmft	
BFG_DATA		Complete as necessary
BFG_JVM_PROPERTIES		Complete as necessary
QMGR	MQPV	
MQ_PATH	/mqm/V8R0M0	
MQ_LANG	E	
Db2_HLQ	SYS2.Db2.V10	
FTE_CONFIG		Used in migration
CREDENTIAL_PATH		Used in migration
DB_PROPS_PATH		Used in migration
BFG_WTO	YES	To get an MFT message on the syslog.
ADMIN_JOB1		Job card. All jobs are generated with the same JCL card.
OUTPUT_CLASS	*	
JOBCARD1		This is the job card for the long running tasks, agents and loggers.
PATH	bin:/usr/bin:/usr/sbin	
armELEMTYPE	If ARM is being used, use the ARM ELEMTYPE specified in the ARM policy. For example, armELEMTYPE=SYSBFGAG for an agent or armELEMTYPE=SYSBFGLG for a logger. If ARM is not being used, set this parameter to blank; for example, armELEMTYPE=	

Table 6. Parameters needed for member BFGCUSTM (continued)

Name	Example data	Comments
armELEMENT	If ARM is being used, use the ARM ELEMENT value specified in the ARM policy for this agent or logger. If ARM is not being used, set this parameter to blank; for example, armELEMENT=	
coordinationQMgr	MQPV	Mandatory configuration

In addition you must review the following variables and supply values where necessary:

- coordinationQMgrHost=
- coordinationQMgrPort=
- coordinationQMgrChannel=
- connectionQMgr=
- connectionQMgrHost=
- connectionQMgrPort=
- connectionQMgrChannel=

These properties are common to the AGENT or LOGGER.

Note: Host, Port, and Channel are required for client connection but should be left blank for a bindings connection on the local machine.

Related concepts

[“Items to check” on page 66](#)

Ensure that you have enough disk space, a directory for storing data, and that the requisite files exist.

[“Editing member BFGCUSTM” on page 69](#)

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

Items to check

Ensure that you have enough disk space, a directory for storing data, and that the requisite files exist.

Check you have enough disk space

Check that you have enough disk space available on the file system where you are going to store the configuration specific files.

If an agent trace is enabled then by default it can use 100 MB of disk space.

The configuration files themselves are small, only a few KB in size.

If you are planning on using two agents and a logger then you need at least 300 MB. You can use the command **df -k path**, where path is the location of the installation specific files. This gives the available and total space in KB.

300 MB is 307,200 KB so you should allow for at least 310,000 KB

Create and check the directory for storing Managed File Transfer data

You need a directory for storing the Managed File Transfer (MFT) data.

Check you have enough space in the file system **df -k /var**. This file system should have at least 310,000 KB available.

If you have not created this file system, use the **mkdir** command; for example **mkdir /var/mft**.

Display what permissions users have on this directory, using the command **ls -ltrd /var/mft**.

If the owner or group is not correct, use the command **chown owner:group /var/mft**.

If permissions for the group are not correct, use the following command to give the owner and the group read, write, and execute permissions. Note that the following command also gives all users read and execute permissions **chmod 775 /var/mft**.

Check the files exist and you have access to them

Use the **ls -ltr** command for the files you will be using during customization. For example:

```
ls -ltrd /java/java71_bit64_GA/J7.1_64/bin
```

gives

```
drwxr-xr-x 4 SYSTASK TSUSER 8192 Nov 15 2013 /java/java71_bit64_GA/J7.1_64/bin
```

where the **drwxr-xr-x** means

d

This is a directory.

rwX

The owner *SYSTASK* has read, write and execute access to the directory.

r-x

People in the group *TSUSER* can read and execute files in the directory.

r-x

Universal access, that is, anyone can read or execute files in the directory.

Check the files specified in:

Path	Access required by users doing the configuration
BFG_JAVA_HOME	Read and execute
/tmp	Read and write
BFG_PROD	Read
BFG_DATA	Write
MQ_PATH	Read

Related concepts

[“Before you start” on page 64](#)

Managed File Transfer (MFT) configuration uses files in UNIX System Services (USS) and PDSE data sets.

[“Common MFT for z/OS configurations” on page 67](#)

An overview of the different Managed File Transfer configurations

Common MFT for z/OS configurations

An overview of the different Managed File Transfer configurations

Managed File Transfer uses agents attached to a queue manager for transferring data.

MFT can use multiple queue managers:

- One or more queue managers to transfer the data.

- A commands queue manager that issues requests. For example, a request to start a transfer is sent to this queue manager, and the associated commands are routed to the MFT agents.
- A coordination queue manager that manages the work.

There are three common Managed File Transfer (MFT) configurations:

1. A single queue manager with one or more agents using local connections. This might be used to put the contents of a data set into IBM MQ queues.
2. A single queue manager with an MFT client on a distributed machine using client bindings.
3. Two queue managers connected by channels, and one or more agents on each machine. These agents can be client or local bindings.

Note the following points:

1. MFT is written in Java, with some shell scripts and JCL to configure and operate MFT.
2. Db2 status and activity can be logged, and this can be stored in Db2 tables.
3. The person configuring MFT must be familiar with Unix System Services (USS). For example:

- The directory structure with files with names like `/u/userID/myfile.txt2`
- USS commands, such as:

cd (change directory)

ls (list)

chmod (change file permissions)

chown (change file ownership or groups that can access the file or directory)

4. The following products are required in USS to be able to configure and run MFT:

- Java; for example, `/java/java71_bit64_GA/J7.1_64/`
- IBM MQ V800, for example `/mqm/V8R0M03`.
- Db2 JDBC libraries, if you want to use Db2 for status and history; for example `/db2/db2v10/jdbc/lib`

You need a coordination queue manager. However, you can use the same queue manager to run agents, to process commands, and for coordination. If you are using multiple queue managers, you must pick one to act as the coordinator.

Check your IBM MQ connectivity

If you have an existing MFT coordinator queue manager, you need connectivity between the queue manager where you are doing the configuration, and the coordinating and command queue managers.

Copy SBFGCMDS to create a JCL library

You need to create a JCL library for each agent and logger, The JCL contains the configuration and jobs used to create and run the agent or logger.

For each agent and logger create a copy of the IBM supplied SBFGCMDS library by editing and running the BFGCOPY member.

This library is used to define the configuration for the agent or logger and, after customization, contains jobs that can be used to create the required Managed File Transfer configuration and agent or logger.

You create member BFGCUSTM as part of this process.

Note: If you are familiar with USS commands, you can configure z/OS with the same commands that you use on other platforms.

Related concepts

[“Common MFT for z/OS configurations” on page 67](#)

An overview of the different Managed File Transfer configurations

[“Editing member BFGCUSTM” on page 69](#)

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

Editing member BFGCUSTM

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

See [Parameters needed for member BFGCUSTM](#), for a list of the parameters requiring specific values.

In addition you must review the following variables and supply values where necessary:

- coordinationQMgrHost=
- coordinationQMgrPort=
- coordinationQMgrChannel=
- connectionQMgr=
- connectionQMgrHost=
- connectionQMgrPort=
- connectionQMgrChannel=

These properties are common to the AGENT or LOGGER.

Note: Host, Port, and Channel are required for client connection but should be left blank for a bindings connection on the local machine.

If this is the first queue manager in your Managed File Transfer environment, and you want to use the same queue manager for coordination, commands, and running agents, set the values to the local queue manager name.

```
coordinationQMgr=MQPV
connectionQMgr=MQPV
```

where MQPV is your local queue manager name.

Submit the job, which updates the PDSE, and creates a directory structure under the specified path.

Note that this job requires exclusive use, so you need to stop using the PSDE while the job runs.

Tip: Whenever you submit job BFGCUSTM, the job replaces all the JCL files. You should rename each member you change.

Related concepts

[“Before you start” on page 64](#)

Managed File Transfer (MFT) configuration uses files in UNIX System Services (USS) and PDSE data sets.

[“Creating an agent” on page 71](#)

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Defining the coordination queue manager

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

Depending on the configuration that you have chosen, this queue manager is on the local MVS system, or on another machine. In the former case, the connections to it are bindings connections and in the latter case, they are client connections.

After you have run the configuration step successfully there are configured members in the PDSE.

Member BFGCFR defines the coordination queue manager, and this job:

1. Creates a directory structure in the Managed File Transfer (MFT) directory, and creates configuration files.
2. Runs CSQUTIL to define IBM MQ resources.

If the coordination queue manager is on a remote machine then this job step fails.

Member BCFCFCR creates files in USS and creates MQ definitions. This job:

1. Creates an MFT topic,
2. Creates an MFT queue
3. Alters `NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)` to be `NAMES(SYSTEM.BROKER.DEFAULT.STREAM, SYSTEM.BROKER.ADMIN.STREAM, SYSTEM.FTE)`
4. Performs `ALTER QMGR PSMODE(ENABLED)`

A `DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)` command is issued before doing the alter. If your NAMLIST is not the default, you should alter your name list to add SYSTEM.FTE to your namelist

Rename member BCFCFCR with your own prefix, for example, CCPCFCR, because re customizing this file replaces it.

Edit this renamed member by inserting the name of your credentials file. For example:

```
%BFGCMD CMD=fteSetupCoordination +
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>'
```

Save and submit the job. Note that if you need to resubmit the job, you need to add the `-f` option.

When this job runs it lists the IBM MQ resources it creates. You need to protect these resources.

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
ALTER TOPIC('SYSTEM.FTE') NPMGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for MFT Pub/Sub interface')
* Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST
* Value prior to alteration:
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +
NAMES(SYSTEM.BROKER.DEFAULT.STREAM+
,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
* Altering PSMODE. Value prior to alteration:
DISPLAY QMGR PSMODE
ALTER QMGR PSMODE(ENABLED)
```

Related tasks

[“Defining the command queue manager” on page 70](#)

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

Defining the command queue manager

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

About this task

You must have a command queue manager, however, you can use the same queue manager for the coordination and command queue managers. Otherwise, you need to create a new command queue manager. This can be on the same machine as the coordination queue manager, but does not have to be.

Procedure

1. Rename member BFGCMCR with your own prefix, for example, CCPCMCR.
You must rename BFGCMCR because re-customizing this file replaces it.

2. Edit the renamed member by inserting the name of your credentials file.

For example:

```
%BFGCMD CMD=fteSetupCommands +  
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>' +
```

3. Save and submit the job.

Note that if you need to resubmit the job, you need to add the *-f* option.

This queue manager is used for commands such as **ftePingAgent**.

4. Review this member, submit it, and review the output.

What to do next

See [“Creating an agent” on page 71](#) for information on how you create an agent.

Related concepts

[“Defining the coordination queue manager” on page 69](#)

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

Related tasks

[“Configuring MQMFTCredentials.xml on multiplatforms” on page 40](#)

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT commands that connect with a queue manager to supply user ID and password credentials. Similarly, MFT loggers might be required to specify a user ID and password when connecting to a database. This credential information can be stored in the MFT credentials file.

Related reference

[“MFT credentials file format” on page 785](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

Creating an agent

You need to copy the PDSE to make the agent-specific PDSE, for example `user.MFT.AGENT1`. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Review member `BFGCUSTM` and if you need to use a different credentials file, create one.

Much of the content remains the same from the customization detailed in [“Editing member BFGCUSTM” on page 69](#).

You need to change:

- `//SYSEXEC DD DSN=SCEN.FTE.JCL.AGENT1`
- `LIBRARY` to match the agent PDSE
- `SERVICE_TYPE=AGENT`
- `NAME` to be the name of the agent (matching the PDSE) `JOBCARD`
- Change `BFG_JVM_PROPERTIES="-Xmx1024M"`

Submit this job, remembering that the job requires exclusive access to the data set.

The jobs for the agent all have names of the form `BFGAG*`

Rename member `BFGAGCR`. This job updates files in the Managed File Transfer directory and uses `CSQUTIL` to create agent specific queues in the local queue manager. Specify the name of your credentials file, for example, `-credentialsFile //' SCEN.FTE.JCL.VB(CREDOLD)`. If you do not specify the name, the job to start the agent does not use a credentials file.

Check the output to ensure that the process has run successfully.

Tip: Copy the path name of the *agent.properties* file from the output of the job to a member in the PDSE for the agent.

For example, copy `/u/userid/fte/wmqmft/mqft/config/MQPA/agents/AGENT1/agent.properties` into member AGENT.

This is useful if you need to display the properties file, and add the line `/u/userid/fte/wmqmft/mqft/logs/MQPA/agents/AGENT1/logs`.

This is where trace files are stored.

Related concepts

[“Defining the coordination queue manager” on page 69](#)

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

[“Using the agent” on page 72](#)

How you use various commands to ensure that the agent is working correctly.

Related tasks

[“Defining the command queue manager” on page 70](#)

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

Using the agent

How you use various commands to ensure that the agent is working correctly.

Start the agent

Rename member BFGAGST, review the member, and submit the job.

If this works you receive message BFGAG0059I: The agent has been successfully started.

Display the active agent(s)

Rename member BFGAGLI, review the member and submit the job which uses the coordinating queue manager.

You must resolve any connectivity problems

Ping the agent to check it is working

Rename member BFGAGPI, review the member and submit the job which uses the command queue manager.

You must resolve any connectivity problems

Carry out a test transfer

See [“Performing a verification transfer” on page 79](#) for further information.

Stop the agent

Rename member BFGAGSP, review the member and submit the job.

Restart the agent using the member BFGAGST.

Related concepts

[“Creating an agent” on page 71](#)

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Updating an existing MFT Agent or Logger command data set on z/OS

You can update a Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

Procedure

1. Edit the BFGCUSTM JCL script member and update variables and properties in the BFGSTDIN DD statement.

If you want to remove a property that was previously defined, set its value to blank, instead of removing the entry. When the BFGCUSTM JCL script is run, the specified properties are applied as an update to the actual agent and logger UNIX System Services properties files; setting a property to a blank value indicates that the property is to be removed.

2. Submit job BFGCUSTM. This job generates the set of JCL commands again, appropriate for the agent or logger. For a full list of the commands, see [“z/OS agent and logger command JCL scripts”](#) on page 78. Examine the output job log to check that the JCL script ran successfully. If there are any failures, correct them and submit the BFGCUSTM job again.

Results

You can modify the generated JCL scripts and add your own logic. However, be careful when you run BFGCUSTM again because you might overwrite the custom logic.

Related concepts

[“MFT configuration options on z/OS”](#) on page 58

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

Related tasks

[“Creating an MFT Agent or Logger command data set”](#) on page 62

You can create a PDSE data set of commands from the Managed File Transfer command template data set for a specific Managed File Transfer Agent or Managed File Transfer Logger for a specific coordination.

z/OS JCL variables

You can use substitution values, JCL variables, and configuration properties in the BFGCUSTM script.

The following table lists the substitution values for the BFGCUSTM JCL script in an MFT command PDSE library data set. You must replace these substitution values with suitable values before you submit the BFGCUSTM job.

Substitution variable	Value
++library++	The data set name of the containing MFT command PDSE library.
++bfg_java_home++	The location of your Java installation.
++bfg_prod++	The location of the MFT product installation UNIX System Services root directory.

The following table describes the environment variables for the BFGSTDIN DD statement for the BFGCUSTM JCL script, in an MFT command PDSE library data set (in the [Variables] section). You must

replace all variables that are specified with substitution values (that is, values enclosed in two plus signs, ++) with suitable values before you submit the BFGCUSTOM job.

<i>Table 9. Environment variables</i>	
Environment variable	Value
LIBRARY	The data set name of the containing MFT command PDSE library.
TMPDIR	UNIX System Services directory for temporary files.
BFG_PROD	The location of the MFT product installation UNIX System Services root directory.
BFG_DATA	The location of the data directory for Managed File Transfer for z/OS, which is the path to <i>DATA_DIR</i> .
BFG_JAVA_HOME	The location of your Java installation.
BFG_JVM_PROPERTIES	Optional. Sets a value for the BFG_JVM_PROPERTIES environment variable. These properties are passed to the Java virtual machine.

Table 9. Environment variables (continued)

Environment variable	Value
BFG_GROUP_NAME	<p>The mqm file group is typically associated with MFT configuration data files and commands. Consequently, all users who are members in the mqm group can access and make changes to the MFT configuration. For more information, see “File system permissions for MFT in IBM MQ” on page 334.</p> <p>For a z/OS system, a file group is a USS filesystem entity, and the mqm file group is not necessarily defined. You can associate a z/OS USS filesystem group for MFT configuration data files by using the BFG_GROUP_NAME environment variable. For example, at the USS shell prompt use:</p> <pre data-bbox="862 680 1472 758">export BFG_GROUP_NAME=FTEGB</pre> <p>which defines a group <i>FTEGB</i> to be associated with any subsequently created configuration files for the current USS session.</p> <p>You can set BFG_GROUP_NAME to a blank value, or remove it.</p> <p>Note: When running BFGCUSTM for the first time, if the MFT configuration is to be used by multiple user ID's, it is important that BFG_GROUP_NAME is set to a group accessible to all required user ID's. If BFGCUSTM is run again, then BFG_GROUP_NAME must not be changed (otherwise, the USS group file permissions for all files and directories in the directory referenced by BFG_DATA must also be changed to reflect the new BFG_GROUP_NAME setting).</p> <p>V9.0.0.1 If you run the fteMigrateAgent command on a z/OS system with the BFG_GROUP_NAME environment variable set to a non-blank value, the command checks to see whether the user is a member of the group named by the BFG_GROUP_NAME variable. If the user is not in the named group, the command might report the error message BFGCL0502E: You are not authorized to perform the requested operation. and fail to run. For details on the criteria the user must meet to successfully run that command, see “fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later” on page 462.</p>

Table 9. Environment variables (continued)

Environment variable	Value
BFG_WTO	z/OS logging is enabled when BFG_WTO is set to YES, ON, or TRUE. This controls whether messages that are written to the agent event log are also written to the z/OS operator log facility, which allows easier access for automation products when you run an agent from JCL. The routing code is Programmer Information (11) and the descriptor code is Informational (12).
SERVICE_TYPE	Specifies whether the MFT command library is for an agent or logger. The valid values are AGENT or LOGGER.
NAME	The name of the agent or logger for the SERVICE_TYPE value.
QMGR	The name of the local queue manager that is associated with the agent or logger for the SERVICE_TYPE value.
OUTPUT_CLASS	The output class for SYSOUT data sets. Defaults to * which requests the same output class as the MSGCLASS parameter from the job statement.
MQ_PATH	Used in BFGPROF to create the LIBPATH environment variable.
MQ_HLQ	The high-level qualifier for IBM MQ data sets.
MQ_LANG	The language that is required.
DB2_HLQ	Optional. High-level qualifier for Db2 data sets.
JOBCARD1	Header line 1 for a JCL command job.
JOBCARD2	Header line 2 for a JCL command job.
JOBCARD3	Header line 3 for a JCL command job.
ADMIN_JOB1	Header line 1 for an admin job.
ADMIN_JOB2	Header line 2 for an admin job.
ADMIN_JOB3	Header line 3 for an admin job.
FTE_CONFIG	Existing WMQFTE configuration for migration. Set to a blank value if migration is not required.
CREDENTIAL_PATH	Path to credentials file for migration, for example /u/user1/agent3. Credentials files for migration for Managed File Transfer must be located in a separate file to configuration information and configuration files on IBM WebSphere MQ File Transfer Edition 7.0.4.4. Required for migration commands BFGAGMG and BFGLGMG JCL scripts only. Set to a blank value if migration is not required.

Table 9. Environment variables (continued)

Environment variable	Value
DB_PROPS_PATH	Specifies the database logger properties file for migration. This option is required only if the properties file does not use the following default name and path: <code>config_directory/coordination_qmgr/databaselogger.properties</code> . Set to a blank value if migration is not required.

Note: The IBM MQ jar files are shipped with MFT, in directory `MQMFT product root/java/lib`, are always used, and not configurable.

The following table describes the mandatory MFT configuration properties for the BFGSTDIN DD statement for the BFGCUSTM JCL script in an MFT command PDSE library data set. You must replace properties specified with substitution values (that is, values enclosed in two plus signs, ++) with a suitable non-blank value before you submit the BFGCUSTM job. These properties define overrides for the MFT configuration properties. You can add agent and logger properties to customize agents or loggers for your environment. For a list of all properties, see “Configuration properties files” on page 85.

Table 10. Mandatory configuration properties for the BFGSTDIN DD statement

Property	Value
coordinationQMGr	The name of the coordination queue manager for the configuration that the agent or logger is associated with.
coordinationQMGrHost	Optional. Host name of the system that the coordination queue manager is running on. If you leave the value for this property blank, a bindings mode connection is assumed.
coordinationQMGrPort	Optional. Port number that the coordination queue manager is listening on. This parameter is used only if you also specify a non-blank value for the coordinationQMGrHost property.
coordinationQMGrChannel	Optional. Channel to use to connect to the coordination queue manager. This parameter is used only if you also specify a non-blank value for the coordinationQMGrHost property.
connectionQMGr	The name of the command queue manager for the configuration that the agent or logger is associated with.
connectionQMGrHost	Optional. Host name of the system that the command queue manager is running on. If you leave the value for this property blank, a bindings mode connection is assumed.
connectionQMGrPort	Optional. Port number that the command queue manager is listening on. This parameter is used only if you also specify a non-blank value for the connectionQMGrHost property.

Table 10. Mandatory configuration properties for the BFGSTDIN DD statement (continued)

Property	Value
connectionQMgrChannel	Optional. Channel to use to connect to the command queue manager. This parameter is used only if you also specify a non-blank value for the connectionQMgrHost property.

z/OS agent and logger command JCL scripts

The set of JCL commands available in an MFT command PDSE library data set.

Table 11. JCL commands available in an MFT command PDSE library data set

Member	Description or fte command line command
BFGCOPY	Job to create a copy of this library
BFGCUSTM	Job to customize this library for agent or logger
BFGCFCR	fteSetupCoordination
BFGCMCR	fteSetupCommands
BFGAGCR	fteCreateAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGCRS	fteCreateLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGAGST	fteStartAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGAGSTP	fteStartAgent procedure. Created only when you set the SERVICE_TYPE variable to AGENT.
BFGAGPI	ftePingAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGAGSP	fteStopAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGST	fteStartLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGLGSTP	fteStartLogger procedure. Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGLGSP	fteStopLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGAGSH	fteShowAgentDetails . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGSH	fteShowLoggerDetails . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGCFDF	fteChangeDefaultConfigurationOptions
BFGAGCL	fteCleanAgent . Created only when you set the SERVICE_TYPE variable to AGENT.

Table 11. JCL commands available in an MFT command PDSE library data set (continued)

Member	Description or fte command line command
BFGAGDE	<code>fteDeleteAgent</code> . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGDE	<code>fteDeleteLogger</code> . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGPRSH	<code>fteDisplayVersion</code>
BFGAGLI	<code>fteListAgents</code> . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGMCLI	<code>fteListMonitors</code>
BFGSTLI	<code>fteListScheduledTransfers</code>
BFGTMLI	<code>fteListTemplates</code>
BFGAGMG	fteMigrateAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGMG	fteMigrateLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGCROBS	fteObfuscate sample
BFGRAS	fteRAS
BFGAGTC	<code>fteSetAgentTraceLevel</code> . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGTC	“ <code>fteSetLoggerTraceLevel</code> ” on page 486. Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGPRANS	fteAnt sample
BFGTRCAS	fteCancelTransfer sample
BFGMNCRS	fteCreateMonitor sample
BFGTMCRS	fteCreateTemplate sample
BFGTRCRS	fteCreateTransfer sample
BFGMNDES	fteDeleteMonitor sample
BFGSTDES	fteDeleteScheduledTransfer sample
BFGTMDES	fteDeleteTemplates sample

z/OS

Note: The JCL, for commands that create MQSC or reference delete scripts, asks you to run a script, but the script has already been run by the job.

z/OS Performing a verification transfer

How you carry out a transfer to check that the product is working correctly.

Rename and edit member BFGTRCRS.

1. Add a /* before the %BFGCMD `CMD=fteCreateTransfer -h`
2. Remove the other comments in the member.

3. Specify the current agent name for -sa and -da
4. Save the JCL
5. Submit the JCL

This JCL connects to the command queue manager.

Configuring a logging task

The logging task needs run on the same image as the coordination queue manager. You can log to Db2.

Creating a logging task

Copy the PDSE to make the logger specific PDSE. For example, user.MFT.LOGGER.

If you need to use a different credentials file, create one. For more information, see [“Configuring MQMFTCredentials.xml on z/OS” on page 42.](#)

Review member `BFGCUSTM`. Note that much of the content remains the same from the previous customization.

However, you need to:

- Change `//SYSEXEC DD DSN=SCEN.FTE.JCL....`
- Change `LIBRARY` to match the agent PDSE
- Change `QMGR` to the name of the coordination queue manager
- Make `SERVICE_TYPE=LOGGER`
- Change `NAME` to be the name of the logger (matching the PDSE)
- Review `JOB CARD` and change the jobname so that the name is different from the job names of the agents.
- Review `BFG_JVM_PROPERTIES="-Xmx1024M"`

If you are using the Db2 logger it is useful to create a file, so that you can capture Db2 traces to help identify Db2 problems.

The name of the file is specified in the JVM properties, where the JDBC trace properties file has content such as

```
db2.jcc.traceDirectory=/u/johndoe/fte
db2.jcc.traceFile=jccTrace1
db2.jcc.traceFileAppend=false
# turn on all traces
# db2.jcc.traceLevel=-1
# turn off all traces
db2.jcc.traceLevel=0
```

Set two JVM properties

```
BFG_JVM_PROPERTIES=-Ddb2.jcc.propertiesFile=/u/.../sql.properties
-Ddb2.jcc.ssid=DBCA
```

Where `/u/.../sql.properties` is the name of your Db2 trace properties file, and `DBCA` is the name of your Db2 subsystem.

Submit this job, noting that the job requires exclusive access to the data set. The jobs for the agent all have names like `BFGLG*`.

Logging to files

For more information on logging to Db2, see [“Creating a logging task, when logging to Db2” on page 81](#)

Rename member BFGLGCRS. This job updates files in the Managed File Transfer (MFT) directory and uses CSQUTIL to create agent specific queues in the local queue manager.

The original file has command %BFGCMD CMD=fteCreateLogger -h which lists the syntax of the command.

To create the logger task comment out the %BFGCMD CMD=fteCreateLogger -h by putting /* in front of the statement, making sure that column one is blank.

Remove the comments from the second command and configure the statements. For example:

```
%BFGCMD CMD=fteCreateLogger  +
-p MQPH      +
-loggerQMgr MQPH      +
-loggerType FILE      +
-fileLoggerMode circular  +
-fileSize 5MB +
-fileCount 5 +
-p MQPH +
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>'
LOGGER
```

Check the output to see that it has processed successfully.

Tip: Copy the pathname of the logger.properties file from the output of the job to a member in the PDSE of the agent.

For example copy into member APATH

```
/u/user_ID/fte/wmqmft/mqft/config/MQPH/loggers/LOGGER/logger.properties
```

This is useful if you need to display properties file.

Add the directory to this file:

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/
```

If you are logging to file, the log files are stored in this directory, for example LOGGER0-20140522123654897.log.

Trace files are in the log subdirectory, for example

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/logs
```

You can now [start the logging task](#).

Creating a logging task, when logging to Db2

Rename member BFGLGCRS.

This job updates files in the MFT directory and uses CSQUTIL to create agent specific queues in the local queue manager.

You need to know:

Db2 name	Example
-dbName databaseName	You can get this from the location value in message DSNL004I for your Db2 subsystem
-dbDriver filePath	For example /db2/db2v10/jdbc/classes/db2jcc.jar
-dbLib filePath	For example /db2/db2v10/jdbc/lib/libdb2jccct2zos_64.so

Edit the file. The original file has command `%BFGCMD CMD=fteCreateLogger -h` which lists the syntax of the command.

Remove the comments from the second command and configure the statements. For example

```
%BFGCMD CMD=fteCreateLogger +
-p MQPH +
-loggerMgr MQPH +
-loggerType DATABASE +
-dbType DB2 +
-dbName DSNDBCP +
-dbDriver /db2/db2v10/jdbc/classes/db2jcc.jar +
-dbLib /db2/db2v10/jdbc/lib/ +
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>' +
LOGGER
```

To create the logger task comment out the `%BFGCMD CMD=fteCreateLogger -h` by putting `/*` in front of the statement, making sure that column one is blank.

Submit the job and check the output to see that it has processed successfully.

Tip: Copy the pathname of the `logger.properties` file from the output of the job to a member in the PDSE of the agents.

For example copy into member APATH:

```
/u/user_ID/fte/wmqmft/mqft/config/MQPH/loggers/LOGGER/logger.properties into member USS
```

This is useful if you need to display the properties file

Trace files are in the log subdirectory, for example:

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/logs
```

Creating Db2 tables

You need to create the Db2 tables. The definitions are in the USS file `mqft/sql/ftelog_tables_zos.sql`.

Create a member Db2 in your PDSE. Edit this member and use the COPY command on the command line. Copy from the USS definitions file.

Because site-specific requirements can vary greatly, this file only specifies the basic structures of the tables, and a tablespace where they will be located.

The tablespace is specified, by the SQL script, to ensure that it is created using a buffer pool with a page size sufficient to hold the largest tables rows possible. Note that attributes such as LOB locations, and so on, are not specified.

Your database administrator might want to modify a copy of this file, to define these performance-related attributes.

This file also assumes a default schema name of FTELOG, default tablespace name of FTELOGTS, and database name of FTELOGDB. You can change these names if you need, to match an existing database and any local naming conventions, by following the process described in the comments at the beginning of the file.

Important: Use online facilities like **SPUFI** to run the commands, because there are comments in the file, and batch programs like **DSNTINAD** do not accept comments.

Starting the logger task

Rename, review and submit the member BFGLGST You should get the message BFGDB0023I: The logger has completed startup activities and is now running.

Logger operations

To display the logger status, Rename, review, and submit member BFGLGSH

To stop the logger, Rename, review, and submit member BFGLGSP.

Environment variables for MFT on z/OS

If you are running commands direct from the USS environment, or your own JCL scripts, after customization and configuration you must set a number of environment variables before running the configuration and administration scripts provided by Managed File Transfer. You must set these variables for each user and in each environment that the scripts will be invoked from.

To avoid conflicts with other products, you can choose to create a `.wmqfterc` script in your home directory. The `.wmqfterc` script is then invoked by each of the Managed File Transfer scripts and you can use this script to provide custom environment settings for Managed File Transfer.

There is also one optional environment variable, `BFG_WTO`, that you can set to send messages to the operator log when running agents from JCL.

Environment variable	Value
BFG_JAVA_HOME	The location of your Java installation. For more information about the levels of Java supported, see IBM MQ system requirements .
BFG_DATA	The location of the data directory for Managed File Transfer for z/OS. This is the path to <code>DATA_DIR</code> .
STEPLIB	Must include the following IBM MQ data sets: <ul style="list-style-type: none">• SCSQAUTH• SCSQANLE• SCSQLOAD If you want to run the database logger component on a z/OS system, STEPLIB must also include the following Db2 data sets in the order shown: <ul style="list-style-type: none">• SDSNEXIT• SDSNL0D2• SDSNLOAD
LIBPATH	Must include the location of your IBM MQJava libraries in z/OS UNIX System Services space (for IBM MQ 8.0, the default is <code>/mqm/V8R0M0/java/lib</code>).

The following is an example `.profile` that correctly configures the environment variables for Managed File Transfer:

```
LIBPATH=/mqm/V8R0M0/java/lib:$LIBPATH
STEPLIB=MQM.V800.SCSQAUTH:MQM.V800.SCSQANLE:MQM.V800.SCSQLOAD
PATH=/u/fteuser/bin:/u/fteuser/J7.0/bin:/bin:/usr/bin:/u/fteuser/extras/bin:/bin:$PATH
BFG_JAVA_HOME=/u/fteuser/J7.0
BFG_DATA=/u/fteuser/DATA_DIR
export PATH LIBPATH STEPLIB BFG_JAVA_HOME BFG_DATA
```

Optionally, you can also set the following environment variables:

Table 14. Optional z/OS environment variable

Environment variable	Value
BFG_WTO	<p>One of the following values will enable BFG_WTO :</p> <ul style="list-style-type: none">• YES• ON• TRUE <p>One of the following values will disable BFG_WTO. These values are not case sensitive.</p> <ul style="list-style-type: none">• NULL• NO• OFF• FALSE <p>Enables z/OS logging. By default, this environment variable is disabled.</p> <p>Messages that are written to the agent event log are also written to the z/OS operator log facility, which allows easier access for automation products when you run an agent from JCL. The routing code is Programmer Information (11) and the descriptor code is Informational (12).</p>

Table 14. Optional z/OS environment variable (continued)

Environment variable	Value
BFG_GROUP_NAME	<p>The mqm file group is typically associated with Managed File Transfer configuration data files and commands. Consequently, all users who are members of the mqm group can access, and make changes to the Managed File Transfer configuration. For more information, see “File system permissions for MFT in IBM MQ” on page 334.</p> <p>For a z/OS system, a file group is a USS filesystem entity, and the mqm file group is not necessarily defined. You can define an alternative, existing z/OS USS filesystem group for Managed File Transfer configuration data files by using the BFG_GROUP_NAME environment variable. For example, at the USS shell prompt:</p> <pre data-bbox="860 751 1463 829">export BFG_GROUP_NAME=FTEGB</pre> <p>which defines group FTEGB to be associated with any subsequently created configuration files for the current USS session.</p> <p>You can set BFG_GROUP_NAME to a blank value, or remove it.</p> <p>V9.0.0.1 If you run the fteMigrateAgent command on a z/OS system with the BFG_GROUP_NAME environment variable set to a non-blank value, the command checks to see whether the user is a member of the group named by the BFG_GROUP_NAME variable. If the user is not in the named group, the command might report the error message BFGCL0502E: You are not authorized to perform the requested operation. and fail to run. For details on the criteria the user must meet to successfully run that command, see “fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later” on page 462.</p>

Configuration properties files

A summary of the properties that are used in Managed File Transfer.

- [“The MFT coordination.properties file”](#) on page 517
- [“The MFT command.properties file”](#) on page 522
- [“The MFT agent.properties file”](#) on page 526
- [Logger configuration properties file](#)

Configuring MFT for the z/OS Automatic Restart Manager (ARM)

Managed File Transfer is an ARM enabled application.

Before you begin

For more information about enabling ARM, and defining ARM policies for your system, see [Using the z/OS Automatic Restart Manager \(ARM\)](#).

If you want to use the MFT DB Logger ability to automatically restart and reconnect to a Db2 database, ARM is the only supported restart manager available.

About this task

Using ARM, agents and loggers can be configured for restart by setting the agent/logger properties `armELEMTYPE`, and `armELEMENT`. Property `armELEMTYPE` defines the type of ARM element and property `armELEMENT` is the name of the element that ARM is to register:

- You can set the agent `ELEMTYPE` to `SYSBFGAG`, and `armELEMENT` can be set to correspond with the agent name.
- You can set the logger `ELEMTYPE` to `SYSBFGLG`, and `armELEMENT` can be set to correspond with the logger name.

Note: Agents and loggers that are configured for restart by ARM can only be successfully run from a batch job or a started task. Attempts to start the agent or logger from the USS command line directly will fail with an ARM error reason code.

Example

The following example of a restart policy defines agent `BFGFT7CAG1` as being dependant on queue manager `FT7C`:

```
RESTART_ORDER
  LEVEL(3)
  ELEMENT_TYPE(SYSBFGAG,SYSBFGLG)

RESTART_GROUP(GROUP7C)
  ELEMENT(SYSMQMGRFT7C)
  ELEMENT(BFGFT7CAG1)
  RESTART_ATTEMPTS(3,300)
```

Using Managed File Transfer for z/OS with the JZOS Java launcher

You can apply the instructions in this topic as an alternative method of using Managed File Transfer in your enterprise, on your IBM MQ for z/OS system.

Overview

Managed File Transfer for z/OS (MFT) uses the standard z/OS installation procedure. An alternative way of running MFT commands is to use JCL and the JZOS Java Launcher.

See [JZOS Batch Launcher and Toolkit](#) for further details.

If your JCL fails to process correctly, see [“Common MFT problems with JZOS”](#) on page 288.

Example JCL for IBM MQ 8.0 and later



Attention: For IBM WebSphere MQ File Transfer Edition 7.0, the parameters begin with `FTE_` instead of `BFG_`.

```
//JOHNDOEA JOB 1,MSGCLASS=H
// JCLLIB ORDER=(SCEN.MFT.JCL) (1)
```

```
// INCLUDE MEMBER=BFGJCL8          (2)
// DD * (2A)
// . ${BFG_PROD}/bin/fteBatch createAgent (3)
export IBM_JAVA_OPTIONS="${BFG_JAVA_OPTIONS} ${BFG_LANG}" (4)
export JZOS_MAIN_ARGS="${BFG_MAIN_ARGS}" (4)
//MAINARGS DD *
-agentName MYAGENT (5)
-f
-agentQMgr MQPD
-p MQPD
/*
```

where:

- (1) is the location of included JCL statements
- (2) Include the specified JCL member from the location in 1)
- (2A) This extends the //STDENV - see below
- (3) This is the command to be executed, without the leading fte prefix
- (4) These lines are required, they set up information for JZOS
- (5) The parameters to the command
- The BFGJCL8 member (you can select your own name) invokes JZOS. This member has the STEPLIB and other JCL needed to run MFT.

Other JCL you need to include

You should include JCL for the IBM MQ for z/OS libraries, and if you are using the Db2 logger, the Db2 libraries.

For example:

```
//WMQFTE EXEC PGM=JVMLDM86,REGION=0M PARM='+T' (1)
//STEPLIB DD DSN=SYS1.SIEALNKE,DISP=SHR (2)
//* MQ libraries
// DD DSN=MQM.V800.SCSQAUTH,DISP=SHR MQ Bindings
// DD DSN=MQM.V800.SCSQANLE,DISP=SHR MQ Bindings
// DD DSN=MQM.V800.SCSQLOAD,DISP=SHR MQ Bindings

//* DB2 libraries
// DD DISP=SHR,DSN=SYS2.DB2.V10.SDSNEXIT.DBCP
// DD DISP=SHR,DSN=SYS2.DB2.V10.SDSNLOAD
// DD DISP=SHR,DSN=SYS2.DB2.V10.SDSNLOAD2
//SYSOUT DD SYSOUT=H
//SYSPRINT DD SYSOUT=H
//STDOUT DD SYSOUT=H
//STDERR DD SYSOUT=H

//STDENV DD DSN=SCEN.MFT.JCL(BFGZENV8),DISP=SHR (3)
```

where:

- (1) is the name of the JZOS program. Look in SYS1.SIEALNKE for the version on your system. Add ,PARM='+T' to give additional diagnostics.
- (2) This is the data set with the JZOS program.
- (3) This is the member name of a shell script. It defines parameters needed by MFT. See [“Shell script to define MFT”](#) on page 87.

It can be any data set and member. It needs to be last in the file because the JCL job extends this. See 2A in [“Example JCL for IBM MQ 8.0 and later”](#) on page 86.

Shell script to define MFT

In the [“Other JCL you need to include”](#) on page 87 example, the member BFGZENV8 is used. This is based on the JZOS profile.

You can use the same configuration file for MFT V8 and IBM WebSphere MQ File Transfer Edition 7.0, with some minor changes. Remember that prior to MFT V8 the parameters begin with FTE. See [“Example file” on page 88](#).

You need to know:

- The location where Java is installed
- The location of the IBM MQ for z/OS Java libraries
- The location of the MFT files
- A user ID needs to be in a specific group to be considered as an IBM MQ for z/OS administrator. You need the name of this group
- If you are not using English for the messages, you need to know which language to specify.

Example file

```
# This is a shell script that configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
# Use PARM='+T' and set -x to debug environment script problems
set -x
# . /etc/profile
#
# Java configuration (including MQ Java interface)
#
export _BPXK_AUTOCVT="ON"
export JAVA_HOME="/java/java71_bit64_sr3_fp30/J7.1_64/"
export PATH="/bin:${JAVA_HOME}/bin/classic/"
LIBPATH="/lib:/usr/lib:${JAVA_HOME}/bin"
LIBPATH="$LIBPATH:${JAVA_HOME}/bin/classic"
LIBPATH=$LIBPATH:"/mqm/V8R0M0/java/lib/"
export LIBPATH

export BFG_JAVA_HOME="${JAVA_HOME}"
export BFG_WTO="YES"
export BFG_GROUP_NAME=MQADM
export BFG_PROD="/HMF8800/"
export BFG_CONFIG="/u/johndoe/fteconfig"
# export BFG_LANG=" -Duser.language=de "
export BFG_LANG=" "
```

where:

export _BPXK_AUTOCVT="ON"

Is required for Unicode conversion

export JAVA_HOME="/java/java71_bit64/J7.1_64/"

Is the location of the Java directory. Specify the name of the path for Java. This directory contains bin and other directories.

export PATH="/bin:\${JAVA_HOME}/bin/classic/"

Sets up the path statement for Java executable statements

LIBPATH="/lib:/usr/lib:\${JAVA_HOME}/bin"

Sets up the library path for the Java executable statements

LIBPATH="\$LIBPATH:\${JAVA_HOME}/bin/classic"

Adds more Java libraries to the LIBPATH statement.

LIBPATH=\$LIBPATH:"/mqm/V8R0M0/java/lib/"

Adds IBM MQ for z/OS libraries in the library path. Specify the name of your IBM MQ for z/OS libraries in USS.

export LIBPATH

Makes the LIBPATH available to JZOS

export BFG_JAVA_HOME="\${JAVA_HOME}"

Sets the BFG_JAVA_HOME to the value of JAVA_HOME specified above

export BFG_WTO="YES"

Setting BFG_WTO to YES causes messages to be displayed on the joblog using WTO

export BFG_GROUP_NAME=MQADM

User IDs, which are a member of the specified group, are considered IBM MQ for z/OS administrators

export BFG_PROD="/HMF8800/"

Is the path where the MFT code is located

export BFG_DATA="/u/johndoe/fteconfig"

Is where the MFT configuration information is stored

export BFG_LANG=" -Duser.language=de "

Is a commented out statement to define the language as German

export BFG_LANG=" "

Specifies the language as the default, English.

The contents of the MFT product in `/lib/messages/BFGNVMessages_*.properties` lists the languages available. The default is to leave the value blank, which means that English is used.

For V7 specify:

```
export FTE_JAVA_HOME="${JAVA_HOME}"
export FTE_WTO="YES"
export FTE_GROUP_NAME=SCENU
export FTE_PROD="" /HMF7100/ "
export FTE_CONFIG="/u/johndoe/fteconfig"
export BFG_LANG=" "
```

Note that the `/u/johndoe/fteconfig` is different from the one in BFG_DATA

Related concepts

[Managed File Transfer for z/OS](#)

Related tasks

[Planning for Managed File Transfer](#)

IBM i **Configuring MFT on IBM i**

To start using Managed File Transfer after you have installed it, you must complete some configuration for your coordination queue manager and agent.

About this task

After you have installed, you must run the configuration scripts provided by Managed File Transfer for new coordination queue managers and new agents before you can use the coordination queue managers and agents to transfer files. You must then start the agents you have created.

Procedure

1. For all new coordination queue managers: run the MQSC commands in the `coordination_qmgr_name.mqsc` file against the coordination queue manager. If the coordination queue manager is not on the same computer as the installation, copy the MQSC script file to the computer where the queue manager is located and then run the script.
 - a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
 - b) Change to the following directory: `/QIBM/UserData/mqm/mqft/config/coordination_qmgr_name`
 - c) Issue the following command, replacing `coordination_qmgr_name` with the name of your queue manager:

```
/QSYS.LIB/QMQM.LIB/RUNMQSC.PGM coordination_qmgr_name < coordination_qmgr_name.mqsc
```

You can configure the coordination queue manager manually instead. For more information, see [“Configuring the coordination queue manager for MFT” on page 92.](#)

2. For all new agents: run the MQSC commands in the `agent_name_create.mqsc` file against the agent queue manager.

If the agent queue manager is not on the same computer as the agent, copy the MQSC script file to the computer where the queue manager is located and then run the script.

- a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
- b) Change to the following directory: `/QIBM/UserData/mqm/mqft/config/agent_qmgr_name/agents`
- c) Issue the following command, replacing `agent_qmgr_name` with the name of your agent queue manager and replacing `agent_name` with the name of your agent:

```
/QSYS.LIB/QMQM.LIB/RUNMQSC.PGM agent_qmgr_name < agent_name_create.mqsc
```

You can configure the agent queue manager manually instead. For more information, see [“Configuring MFT Agent queue managers” on page 93.](#)

3. If you have not already started the QMFT subsystem as part of the installation, from the IBM i command line, start the QMFT subsystem using the following command: `STRSBS SBSDB(QMQMMFT/QMFT)`, or `STRSBS QMQMMFT/QMFT`
4. Start your new agents using the **fteStartAgent** command.
 - a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
 - b) Change to the following directory: `/QIBM/ProdData/mqm/bin`
 - c) Issue the following command, replacing AGENT with the name of your agent:

```
./fteStartAgent AGENT
```

What to do next

You are recommended to set up sandboxes to limit the areas of the file system that an agent can access. This feature is described in [“Working with MFT agent sandboxes” on page 45.](#)

Related concepts

[“Configuring MFT for first use” on page 90](#)

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Configuring MFT for first use

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Connecting to IBM MQ

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

Managed File Transfer and channels

Managed File Transfer uses MQI channels to connect agents in client mode to their agent queue managers, and to connect command applications (for example, **fteCreateTransfer**) to their command and coordination queue managers. In the default configuration, these connections are made using a SVRCONN channel called `SYSTEM.DEF.SVRCONN`, which exists by default on all queue managers.

Because of these defaults, you do not need to alter any MQI channels for a basic Managed File Transfer installation.

There are six types of message channel end points, but this topic covers only sender-receiver pairs. See [Distributed queuing components](#) for information about other channel combinations.

Required message paths

IBM MQ messages can travel over message channels only, so you must ensure that channels are available for all message paths required by Managed File Transfer. These paths do not need to be direct; messages can travel through intermediate queue managers if required. This topic covers only direct point-to-point communication. See [How to get to the remote queue manager](#) for more information about these options.

The communication paths used by Managed File Transfer are as follows:

Agent to agent

Any two agents that files are transferred between require bidirectional communication between their associated queue managers. Because this path carries the bulk data, consider making the path as short, fast, or cheap as possible according to your needs.

Agent to coordination

Log messages from the agents that participate in a transfer must be able to reach the coordination queue manager.

Command to agent

Any queue manager that command applications or the IBM MQ Explorer (using the command queue manager) connect to must be able to send messages to the queue managers of the agents that those command applications are used to control. To enable feedback messages to be shown by the commands, use a bidirectional connection.

For more information, see [Verifying an IBM MQ installation](#) for the platform, or platforms, that your enterprise use.

Related concepts

[“Configuring a multi-instance queue manager to work with MFT” on page 97](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring MFT network queue managers” on page 91](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring the coordination queue manager for MFT” on page 92](#)

After running the **fteSetupCoordination** command, run the `coordination_qmgr_name.mqsc` script in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Configuring MFT network queue managers

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

About this task

There are two ways to configure your queue managers to be able to communicate with each other:

- By setting up an IBM MQ queue manager cluster.

For information about IBM MQ queue manager clusters and how to configure them, see [Configuring a queue manager cluster](#).

- By setting up channels between the queue managers, which is described as follows:

Setting up channels between queue managers

Set up the following message channels between your queue managers:

- From the agent queue manager to the coordination queue manager
- From the command queue manager to the agent queue manager.
- From the agent queue manager to the command queue manager (to enable feedback messages to be shown by the commands).
- From the command queue manager to the coordination queue manager
- From the agent queue manager to any other agent queue manager in the Managed File Transfer network

If you need further information about how to set up this communication, start with this information: [Administering remote IBM MQ objects using MQSC](#).

Some suggested example steps are:

Procedure

1. Create a transmission queue on the IBM MQ queue manager with the same name as the coordination queue manager.

You can use the following MQSC command:

```
DEFINE QLOCAL(coordination-qmgr-name) USAGE(XMITQ)
```

2. On the IBM MQ queue manager, create a sender channel to the Managed File Transfer coordination queue manager. The name of the transmission queue created in the previous step is a required parameter for this channel. If communication with Managed File Transfer in IBM WebSphere MQ 7.5 or Managed File Transfer agents is required, ensure the CONVERT parameter of the sender channel is set to no. (Earlier versions of IBM WebSphere MQ File Transfer Edition always published messages in UTF-8 format, which means that any data conversion corrupts the message. This is not necessary for agents on Managed File Transfer in IBM MQ 8.0 or later, as messages are published with a blank format.)

You can use the following MQSC command:

```
DEFINE CHANNEL(channel-name) CHLTYPE(SDR) CONNAME('coordination-qmgr-host(coordination-qmgr-port)')
XMITQ(coordination-qmgr-name) CONVERT(NO)
```

Note: Set CONVERT(NO), only if required.

3. On the Managed File Transfer coordination queue manager, create a receiver channel to the IBM MQ queue manager. Give this receiver channel the same name as the sender channel on the IBM MQ queue manager.

You can use the following MQSC command:

```
DEFINE CHANNEL(channel-name) CHLTYPE(RCVR)
```

What to do next

Next, follow the configuration steps for your coordination queue manager: [Configuring the coordination queue manager](#).

Configuring the coordination queue manager for MFT

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the

necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

About this task

Procedure

1. Create a local queue named SYSTEM.FTE.
2. Add the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist.
3. Create a topic named SYSTEM.FTE with a topic string of SYSTEM.FTE.
4. Ensure the Non-persistent message delivery (NPMSGDLV) and Persistent messages delivery (PMSGDLV) attributes of the SYSTEM.FTE topic are set to ALLAVAIL.
5. Ensure the Publish/Subscribe mode (PSMODE) attribute of the coordination queue manager is set to ENABLED.

What to do next

If you run the `stimqm -c` command on a queue manager that has been configured as a coordination queue manager, the command deletes the change made in [step 2](#) (adding the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist). This is because `stimqm -c` re-creates the default IBM MQ objects and reverses the Managed File Transfer changes. Therefore, if you have started the queue manager with `stimqm -c`, complete either one of the following steps:

- Run the `coordination_qmgr_name.mqsc` script on the queue manager again.
- Repeat [step 2](#).

Related concepts

[“Connecting to IBM MQ” on page 90](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

[“Configuring a multi-instance queue manager to work with MFT” on page 97](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring MFT network queue managers” on page 91](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

Related reference

[“fteSetupCoordination” on page 492](#)

The **`fteSetupCoordination`** command creates properties files and the coordination queue manager directory for Managed File Transfer.

Configuring MFT Agent queue managers

After installation, run the `agent_name_create.mqsc` script in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory to perform the necessary configuration for the agent queue manager. However, if you want to do this configuration manually, complete these steps on the agent queue manager.

Procedure

1. Create the agent operation queues.

These queues are named:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

For information about the queue parameters, see [“MFT Agent queue settings” on page 595](#).

2. Create the agent authority queues.

These queues are named:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

For information about the queue parameters, see [“MFT Agent queue settings” on page 595](#).

What to do next

For information about creating and configuring a protocol bridge agent, see [“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)” on page 375](#) and [“Configuring a protocol bridge for an FTPS server” on page 219](#).

Creating an IBM MQ File Transfer structure

You can configure a Managed File Transfer structure, based on a single agent connected to a queue manager on the same machine.

About this task

The MFT configuration is stored in a file structure under the IBM MQ DataPath, on the machine that the agent will be located.

The following sample configuration is for an MFT IBM MQ 8.0 queue manager named SAMPLECOORD (with security disabled), and a single MFT agent named SAMPLEAGENT:

```
+--- config
      +--- SAMPLECOORD
            +--- command.properties
            +--- coordination.properties
            +--- SAMPLECOORD.mqsc
            +--- agents
                  +--- SAMPLEAGENT
                        +--- agent.properties
                        +--- SAMPLEAGENT_create.mqsc
                        +--- SAMPLEAGENT_delete.mqsc

+--- logs
      +--- SAMPLECOORD
            +--- agents
                  +--- SAMPLEAGENT
                        +--- logs
```

This example assumes that queue manager security has been disabled. The following commands, run in **runmqsc**, will disable security after the queue manager is restarted:

```
runmqsc queue manager
```

```
alter qmgr CONNAUTH(NONE);
alter qmgr CHLAUTH(DISABLED);
end;
```

For configuration with security enabled in MFT in IBM MQ 8.0 or later, **CONNAUTH** requires all MFT commands that connect with a queue manager to supply user ID and password credentials. You can apply the additional parameters **-mquserid** and **-mqpassword** for each command, or define a `MQMFTCredentials.xml` file. The following sample credential file defines the user ID of `fteuser`, for which the password of `MyPassword` is to be used when connecting to the queue manager `SAMPLECOORD`:

```
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/MQMFTCredentials MQMFTCredentials.xsd">
  <tns:qmgr mqPassword="MyPassword" MyUserId="fteuser" name="SAMPLECOORD"/>
</tns:mqmftCredentials>
```

For more information, see [“MFT and IBM MQ connection authentication”](#) on page 39.

Notes:

- To locate your MFT configuration directory, use the **`fteDisplayVersion -v`** command.
- For z/OS users, the `MQMFTCredential.xml` file can be located as a member in a partitioned data set with variable record format (RECFM=V), or undefined record format (RECFM=U)..
- For configuration with security enabled, add the following parameter to the steps below to associate the credentials with the relevant queue manager: `-credentialsFile full credential file path`.
- The clear text password in the `MQMFTCredential.xml` can be obfuscated using the following command:

```
fteObfuscate -credentialsFile full file path to MQMFTCredentials.xml
```

Procedure

1. Create a coordination queue manager.

A coordination queue manager is a single queue manager, used to receive all transfer log and status information from its agents. Run the following command:

```
fteSetupCoordination -coordinationQMgr coordination_qmgr_name
```

This creates the basic top level configuration, and create an IBM MQ script file to call `coordination_qmgr_name.mqsc`.

The configuration then needs to be loaded into the queue manager, by running the following IBM MQ command:

```
runmqsc queue manager name < coordination_qmgr_name.mqsc
```

Note: For TCP client connection to a queue manager, you can use:

```
fteSetupCoordination -coordinationQMgr coordination_qmgr_name
-coordinationQMGrHost coordination_qmgr_host -coordinationQMGrPort coordination_qmgr_port
-coordinationQMGrChannel coordination_qmgr_channel
```

For the created `coordination_qmgr_name.mqsc`, you will need to run the **`runmqsc`** command on the same machine that the coordination queue manager is running on.

2. Create the command queue manager.

A command queue manager is a single queue manager which has been pre-configured so that the IBM MQ infrastructure can route MFT requests to the relevant agent. Run the following command:

```
fteSetupCommands -connectionQMGr Command QM Name -p Coordination QM Name
```

This creates a `command.properties` file in the coordination directory. Note that the `-p` is optional, and not required if the commands are being set up for the default coordination.

Note: For TCP client connection to a queue manager, you can use:

```
fteSetupCommands -p coordination_qmgr_name -commandQMGr connection_qmgr_name  
-commandQMGrHost connection_qmgr_host -commandQMGrPort connection_qmgr_port  
-commandQMGrChannel connection_qmgr_channel
```

3. Create the agent.

An agent is an application that can send and receive files. Run the following command:

```
fteCreateAgent -p coordination_qmgr_name -agentName agent_name -agentQMGr agent_qmgr_name
```

This creates the agent configuration under the coordination, and creates an IBM MQ script file to call `agent_name.mqsc` in the agent's configuration directory.

Run the following IBM MQ command to load the IBM MQ script file into the queue manager:

```
runmqsc agent_qmgr_name < agent_name_create.mqsc file
```

Note: For TCP client connection to a queue manager, you can use:

```
fteCreateAgent -p coordination_qmgr_name -agentName agent_name -agentQMGr agent_qmgr_name  
-agentQMGrHost agent_qmgr_host -agentQMGrPort agent_qmgr_port -agentQMGrChannel  
agent_qmgr_channel
```

4. Start the agent.

Run the following command:

```
fteStartAgent -p coordination_qmgr_name agentName
```

The agent starts in the background, and the command prompt is returned. To check that the agent is running, run the following command:

```
ftelistAgents -p coordination_qmgr_name
```

This shows the status of the agents. If the agent is running successfully, it is reported as in the READY state.

Results

A basic MFT infrastructure is ready to use, and you can now use the **fteCreateTransfer** command to request a transfer. Alternatively if the IBM MQ Explorer is available, use the MFT plug-ins to create and monitor transfers.

More agents can be added to the configuration by repeating the Step 3: Create the agent. If the TCP client connection is used, these can be on different machines. For different machines the **fteSetupCoordination** and **fteSetupCommands** commands must be repeated for each machine, however the mqsc scripts would not need to be run.

More complex configurations can have separate queue managers for coordination and each agent. In these cases the various queue managers will need to be connected together.

Related reference

[“fteSetupCoordination” on page 492](#)

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for Managed File Transfer.

[“fteSetupCommands: create the MFT command.properties file” on page 490](#)

The **fteSetupCommands** command creates the Managed File Transfer `command.properties` file. This properties file specifies the details of the queue manager that connects to the IBM MQ network when you issue commands.

[“fteCreateAgent \(create an MFT agent\)” on page 371](#)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

[“fteObfuscate: encrypt sensitive data” on page 473](#)

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file.

[“MFT credentials file format” on page 785](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

[“What to do if your MFT agent is not listed by the fteListAgents command” on page 289](#)

If your agent is not listed by the **fteListAgents** command or is not displayed in the IBM MQ Explorer, or your file transfers are not displayed in the **Transfer Log** of the IBM MQ Explorer, you can carry out a number of problem determination steps to investigate the cause.

Configuring a multi-instance queue manager to work with MFT

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Configuring a multi-instance queue manager

Important: For information about configuring an IBM MQ multi-instance queue manager, see [Multi-instance queue managers](#). Ensure that you have read this information before attempting to configure a multi-instance queue manager to work with Managed File Transfer.

Using a multi-instance queue manager as an agent queue manager

To enable an agent to connect to both the active and standby instance of your multi-instance queue manager, add the `agentQMgrStandby` property to the agent's `agent.properties` file. The `agentQMgrStandby` property defines the host name and the port number used for client connections for the standby queue manager instance. The value of the property must be given in MQ CONNAME format, that is, `host_name(port_number)`.

The `agentQMgr` property specifies the name of the multi-instance queue manager. The `agentQMgrHost` property specifies host name for the active queue manager instance and the `agentQMgrPort` property specifies the port number for the active queue manager instance. The agent must connect in client mode to both the active and the standby instance of the multi-instance queue manager.

See [“The MFT agent.properties file” on page 526](#) for more information.

This example shows the contents of the `agent.properties` file for AGENT1 that connects to a multi-instance queue manager called QM_JUPITER. The active instance of QM_JUPITER is on the system host1

and uses the port number 1414 for client connections. The standby instance of QM_JUPITER is on the system host2 and uses port number 1414 for client connections.

```
agentName=AGENT1
agentDesc=
agentQMgr=QM_JUPITER
agentQMgrPort=1414
agentQMgrHost=host1
agentQMgrChannel=SYSTEM.DEF.SVRCONN
agentQMgrStandby=host2(1414)
```

Using a multi-instance queue manager as the coordination queue manager

To enable connections to both the active and standby instance of your multi-instance coordination queue manager, add the `coordinationQMgrStandby` property to all the `coordination.properties` files in your Managed File Transfer topology.

See [“The MFT coordination.properties file” on page 517](#) for more information.

This example shows the contents of a `coordination.properties` file that specifies the connection details to a multi-instance coordination queue manager called QM_SATURN. The active instance of QM_SATURN is on the system `coordination_host1` and uses the port number 1420 for client connections. The standby instance of QM_SATURN is on the system `coordination_host2` and uses the port number 1420 for client connections.

```
coordinationQMgr=QM_SATURN
coordinationQMgrHost=coordination_host1
coordinationQMgrPort=1420
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
coordinationQMgrStandby=coordination_host2(1420)
```

The Managed File Transfer stand-alone logger must always connect to its queue manager in bindings mode. When using the stand-alone logger with a multi-instance coordination queue manager connect stand-alone logger, in bindings mode, to a different queue manager. The steps to do this are described in [“Alternative configurations for an MFT stand-alone logger” on page 120](#). You must define the channels between the stand-alone logger's queue manager and the coordination queue manager with the host name and port number of both instances of the multi-instance coordination queue manager. For information on how to do this, see [Multi-instance queue managers](#).

The Managed File Transfer plug-in for IBM MQ Explorer connects to the coordination queue manager in client mode. If the active instance of the multi-instance coordination queue manager fails the standby instance of the coordination queue manager becomes active and the plug-in reconnects.

The Managed File Transfer commands **`fteList*`** and **`fteShowAgentDetails`** connect directly to the coordination queue manager. If the active instance of the multi-instance coordination is unavailable these commands will attempt to connect to the standby instance of the coordination queue manager.

Using a multi-instance queue manager as the command queue manager

To enable connections to both the active and standby instance of your multi-instance command queue manager, add the `connectionQMgrStandby` property to all the `command.properties` files in your Managed File Transfer topology.

See [“The MFT command.properties file” on page 522](#) for more information.

This example shows the contents of a `command.properties` file that specifies the connection details to a multi-instance command queue manager called QM_MARS. The active instance of QM_MARS is on the system `command_host1` and uses the port number 1424 for client connections. The standby instance of QM_MARS is on the system `command_host2` and uses the port number 1424 for client connections.

```
connectionQMgr=QM_SATURN
connectionQMgrHost=command_host1
```

```
connectionQMgrPort=1424
connectionQMgrChannel=SYSTEM.DEF.SVRCONN
connectionQMgrStandby=command_host2(1424)
```

Related concepts

[“Connecting to IBM MQ” on page 90](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

Related tasks

[“Configuring MFT network queue managers” on page 91](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring the coordination queue manager for MFT” on page 92](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Retaining MFT log messages

Managed File Transfer sends file transfer progress and log information to the coordination queue manager. The coordination queue manager publishes this information to any matching subscriptions to the SYSTEM.FTE topic. If there are no subscriptions, this information is not retained.

If transfer progress or log information is significant to your business, you must take one of the following steps to ensure that the information is retained:

- Use the Managed File Transfer database logger to copy messages published to the SYSTEM.FTE/Log topic to an Oracle or Db2 database.
- Define a subscription to the SYSTEM.FTE topic, which stores publications on an IBM MQ queue. Define this subscription before transferring any file transfers to ensure that all progress and log messages are retained on the queue.
- Write an application that uses the message queue interface (MQI) or IBM MQ JMS to create a durable subscription and process the publications that are delivered to the subscription. This application must be in operation before any files are transferred to ensure that the application receives all progress and log messages.

Each of these approaches is described in more detail in the sections that follow.

Do not rely on the IBM MQ Explorer plug-in to retain log information.

Using the Managed File Transfer database logger to retain log messages

The database logger is an optional component of Managed File Transfer that you can use to copy log information in to a database for analysis and auditing purposes. The database logger is a stand-alone Java application that you install on a system that hosts the coordination queue manager and the database. For more information about the database logger, see [“Configuring an MFT logger” on page 100](#).

Retaining progress and log messages by using the IBM MQ Explorer plugin

When an instance of the IBM MQ Explorer plug-in is first started, the instance creates a durable subscription on the coordination queue manager. This durable subscription is used to collect the information that is displayed in the **Transfer Log** and **Current Transfer Progress** views.

The name of the durable subscription is prefixed to show that the subscription was created by the IBM MQ Explorer MFT plugin, the host name, and the name of the user, for example *MQExplorer_MFT_Plugin_HOST_TJWatson*.

This prefix is added in case an administrator wants to delete a durable subscription that is no longer in active use by an instance of the IBM MQ Explorer plugin.

Using a durable subscription on the coordination queue manager can cause messages to build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume Managed File Transfer network, use the IBM MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

To stop this from happening, you can specify that the IBM MQ Explorer plug-in use a non-durable subscription to the coordination queue manager. Perform the following steps in your IBM MQ Explorer:

1. Select **Window > Preferences > MQ Explorer > Managed File Transfer**
2. From the **Transfer Log subscription type** list, choose NON_DURABLE.

Storing publications on an IBM MQ queue

To store log or progress messages on an IBM MQ queue, configure a subscription on the coordination queue manager that forwards messages to this queue. For example, to forward all log messages to a queue named LOG.QUEUE, submit the following MQSC command:

```
define sub(MY.SUB) TOPICSTR('Log/#') TOPICOBJ(SYSTEM.FTE) DEST(LOG.QUEUE)WSHEMA(TOPIC)
```

After the log messages have been forwarded to an IBM MQ queue, they are persisted on the queue until they are processed by an IBM MQ application that uses the queue.

Writing applications that manage a durable subscription to the SYSTEM.FTE topic

You can write applications that manage their own durable subscriptions to the SYSTEM.FTE topic by using one of the application programming interfaces supported by IBM MQ. These applications can receive IBM MQ queue or log messages and act on them appropriately for your business needs.

For more information about the available application programming interfaces, see [Developing applications](#).

Configuring an MFT logger

When Managed File Transfer transfers files, it publishes information about its actions to a topic on the coordination queue manager. The database logger is an optional component of Managed File Transfer that you can use to copy this information into a database for analysis and auditing purposes.

There are three versions of the logger:

- stand-alone file logger
- stand-alone database logger
- Java Platform, Enterprise Edition (Java EE) logger

Important: Managed File Transfer loggers are not supported on the IBM i platform.

Stand-alone file logger

The stand-alone file logger is a Java process that either runs on the system that hosts the coordination queue manager, or on a system which hosts a queue manager with connectivity to the coordination queue manager. The stand-alone file logger uses IBM MQ bindings to connect to its associated queue manager. The stand-alone logger is created using the **fteCreateLogger** command.

You can run the stand-alone file logger as a Windows service to ensure that the file logger continues running when you log off from your Windows session, and it can be configured to start automatically when a system restarts. For instructions, see [“Installing the MFT stand-alone file logger” on page 101](#).

The stand-alone file logger is not supported on z/OS or IBM i.

Stand-alone database logger

The stand-alone database logger is a Java application that you install on a system that hosts a queue manager and a database. The stand-alone database logger is often installed on the same system as the coordination queue manager, however it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone database logger uses IBM MQ bindings to connect to its associated queue manager, and a type 2 or type 4 JDBC driver to connect to a Db2 or Oracle database. These types of connection are required because the stand-alone database logger uses the queue manager's XA support to coordinate a global transaction over both the queue manager and database, protecting the data.

If you are using a Windows system, you can run the stand-alone loggers as Windows services to ensure that the loggers continue running when you log off from your Windows session. For instructions, see [“Installing the MFT stand-alone database logger” on page 108](#) for a stand-alone database logger.

Java EE database logger

The Java EE database logger is provided as an EAR file, which you install into an application server. This can be more convenient than using the stand-alone database logger if you have an existing Java EE application server environment available because the Java EE database logger can be managed alongside your other enterprise applications. You can also install the Java EE database logger on a separate system to the systems hosting your IBM MQ server and database. The Java EE database logger is supported for use with Db2 and Oracle databases. The Java EE database logger also supports Oracle Real Application Clusters when installed on WebSphere Application Server 7.0.

For instructions on how to configure a logger, see the following topics:

- [“Installing the MFT stand-alone file logger” on page 101](#)
- [“Installing the MFT stand-alone database logger” on page 108](#)
- [“Installing the Java EE database logger for MFT” on page 121](#)

Installing the MFT stand-alone file logger

The stand-alone file logger is a Java process that must connect to a coordination queue manager by using IBM MQ bindings mode or client mode. To define a stand-alone file logger use the **fteCreateLogger** command and follow the steps in this topic.

About this task

For more information about the stand-alone file logger, see [“Configuring an MFT logger” on page 100](#). The steps in this topic configure a logger to connect to a coordination queue manager. For alternative logger configurations see [“Alternative configurations for an MFT stand-alone logger” on page 120](#)

The stand-alone file logger is not supported on z/OS or IBM i.

Procedure

1. Ensure that you have the Managed File Transfer Logger component installed. For more information, see [Managed File Transfer product options](#)
2. Run the **fteCreateLogger** command specifying the coordination queue manager, and setting the parameter `-loggerType` to `FILE` to create your stand-alone file logger. For more information, see [“fteCreateLogger \(create an MFT file or database logger\)” on page 389](#).
3. Optional: If you want to use a custom format then you can modify the XML file created by the **fteCreateLogger** command. The log format definition is located in the `FileLoggerFormat.xml` file. For more information, see [“MFT stand-alone file logger format” on page 102](#).
4. Run the MQSC commands, provided by the **fteCreateLogger** command, against your coordination queue manager to create the logger queues.
5. Identify a user to run the logger process and configure permissions for that user. For more information, see [“Configuring user access for an MFT stand-alone file logger” on page 107](#).

6. Optional: You can configure the stand-alone file logger further by editing the `logger.properties` file created when you ran the **fteCreateLogger** command. This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information about available properties and their affects, see [“MFT logger configuration properties” on page 111.](#)

7. **Windows**

Optional: If you are using a Windows system, you can run the stand-alone file logger as a Windows service. Run the **fteModifyLogger** command with the **-s** parameter. For more information, see [“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471.](#)

8. Start the stand-alone file logger with the **fteStartLogger** command. For more information, see [“fteStartLogger” on page 507.](#)

If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone file logger starts as a Windows service.

9. Check the logger output. The stand-alone file logger generates two types of output, file transfer audit data and logger diagnostic data. The file transfer audit data can be found in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs`. The logger diagnostic data can be found in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name`

10. You can stop the logger by using the **fteStopLogger** command. For more information, see [“fteStopLogger” on page 510.](#)

Results

MFT stand-alone file logger format

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

The configuration directory for the logger is located in `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name`. When creating a new file logger, a version of this file is created that contains a default set of definitions used by the file logger. For more information about the default log format definition, see [“MFT stand-alone file logger default log format” on page 587.](#)

If you want to specify your own custom log format, edit the `FileLoggerFormat.xml` file.

A custom log format definition

A log format definition consists of a set of message types with each message type having a format definition. A format definition for a message type consists of a set of inserts provided in XPATH format and a separator that is used to separate each insert. The ordering of the inserts determines the order in which the content is placed in the lines generated for output to the log files. For example, this is the definition for the `callStarted` message type:

```
<callStarted>
  <format>
    <inserts>
      <insert type="user" width="19" ignoreNull="false">/transaction/action/
        @time</insert>
      <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system" width="6" ignoreNull="false">type</insert>
      <insert type="user" width="0" ignoreNull="false">/transaction/agent/
        @agent</insert>
      <insert type="user" width="0" ignoreNull="false">/transaction/agent/@QMGr</insert>
      <insert type="user" width="0" ignoreNull="false">/transaction/job/name</insert>
      <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/
        call/command/@type</insert>
      <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/
        call/command/@name</insert>
      <insert type="system" width="0" ignoreNull="true">callArguments</insert>
    </inserts>
    <separator></separator>
  </format>
</callStarted>
```

This format produces a line in the log file like this:

```
2011-11-25T10:53:04;414d5120514d5f67627468696e6b20206466cf4e20004f02; [CSTR];  
AGENT1;AGENT_QM;Managed Call;executable;echo;call test;
```

The inserts provided in the format definition are in the order in which the information appears on the line in the log file. For more information on the XML schema defining the format for the `FileLoggerFormat.xml` file, see [“Stand-alone file logger format XSD” on page 591](#).

Message types

The FTE agents write a range of different message types to the `SYSTEM.FTE/Log` sub-topic. For more information, see [“SYSTEM.FTE topic” on page 594](#). The log file definition can contain format definitions for these types of messages:

```
callCompleted  
callStarted  
monitorAction  
monitorCreate  
monitorFired  
notAuthorized  
scheduleDelete  
scheduleExpire  
scheduleSkipped  
scheduleSubmitInfo  
scheduleSubmitTransfer  
scheduleSubmitTransferSet  
transferStarted  
transferCancelled  
transferComplete  
transferDelete  
transferProgress
```

The format of the messages can vary. The majority of message types write a single line in the log file for each log message consumed from the `SYSTEM.FTE/Log` sub-topic. This leads to the simple case where the XPATH addresses provided in the log format definition relate to the root of the message. These are the message types that use this method to write output:

```
callCompleted  
callStarted  
monitorAction  
monitorCreate  
monitorFired  
notAuthorized  
scheduleDelete  
scheduleExpire  
scheduleSkipped  
scheduleSubmitInfo  
scheduleSubmitTransfer  
transferStarted  
transferCancelled  
transferComplete  
transferDelete
```

The other method used to write a log message uses multiple lines to represent the items in a transfer set within a log message. In this case the format provided is applied to each item in the transfer set within the log message. If you want to include information that is specific to each item within the transfer set, then the XPATH provided is required to use the item as its XPATH root. These are the message types that use this method to write output:

```
scheduleSubmitTransferSet  
transferProgress
```

A line of output is written for each item in the transfer set. Information that you want to be fixed for all items in a transfer set can still use XPATH addresses relative to the root of the log message. In the

following simplified transferProgress format definition example it's the timestamp and transfer ID that are fixed. Any information that is relative to an item as its root will vary for each line written. In this example the source and destination file information for each item are written.

```
<transferProgress>
  <format>
    <inserts>
      <insert type="user" width="19" ignoreNull="false">/transaction/action/
        @time</insert>
      <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system" width="6" ignoreNull="false">type</insert>
      <insert type="user" width="3" ignoreNull="true">status/@resultCode</insert>
      <insert type="user" width="0" ignoreNull="false">source/file |
        source/queue</insert>
      <insert type="user" width="0" ignoreNull="false">source/file/@size |
        source/queue/@size</insert>
      <insert type="user" width="5" ignoreNull="true">source/@type</insert>
      <insert type="user" width="6" ignoreNull="true">source/@disposition</insert>
      <insert type="user" width="0" ignoreNull="false">destination/file |
        destination/queue</insert>
      <insert type="user" width="0" ignoreNull="false">destination/file/@size |
        destination/queue/@size</insert>
      <insert type="user" width="5" ignoreNull="true">destination/@type</insert>
      <insert type="user" width="9" ignoreNull="true">destination/@exist</insert>
      <insert type="user" width="0" ignoreNull="true">status/supplement</insert>
    </inserts>
    <separator></separator>
  </format>
</transferProgress>
```

This produces a log file entry of one or more lines in this format:

```
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702; [TPRO];0
; /src/test1.file;3575;file;leave ; /dest/test1.file;3575;file;overwrite;;
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702; [TPRO];0
; /src/test2.file;3575;file;leave ; /dest/test2.file;3575;file;overwrite;;
```

Insert Format

There are two types of insert available when defining a format for a message type: user and system. The type of an insert is defined in the type attribute of the insert element. Both types of inserts can also have their layout customized using the **width** and **ignoreNull** attributes of the insert element. For example:

```
<insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
```

In this example, the insert takes the information found in the log message at /transaction/@ID and trims or pads it to 48 characters before writing it to the log. If the content of /transaction/@ID is null it writes the string null after padding it to 48 characters because the ignoreNull attribute is set to false. If ignoreNull is set to true the empty string, padded to 48 characters, is written instead. Setting width="0" means the column width is not trimmed, it does not mean that the width is trimmed to 0. The ignoreNull attribute can be used in this way to detect in the log when a null is found when it was not expected. This can be useful when debugging a new log file definition.

User defined inserts

A user insert contains an XPATH address for the information to be written in that insert. This address refers to a piece of information found in the FTE log message. For more information about log message formats, see:

- [“File transfer log message formats” on page 723](#)
- [“Scheduled file transfer log message formats” on page 744](#)
- [“MFT monitor log message format” on page 749](#)

System defined inserts

System defined inserts contain a keyword that refers to a piece of information that either cannot be found in the log message or is not easy to define using the XPATH language.

Supported systems inserts are:

- `type` - Writes the type of the log message in a short format.
- `callArguments` - Writes the set of arguments supplied to a managed call in a space separated format.
- `transferMetaData` - Writes the set of metadata entries defined for a transfer in a comma separated `key=value` format.

The following table lists the value of "type" for system defined inserts for each message type.

Message type	Value of "type" system insert
callCompleted	[CCOM]
callStarted	[CSTR]
monitorAction	[MACT]
monitorCreate	[MCRT]
monitorFired	[MFIR]
notAuthorized	[AUTH]
scheduleDelete	[SDEL]
scheduleExpire	[SEXP]
scheduleSkipped	[SSKP]
scheduleSubmitInfo	[SSIN]
scheduleSubmitTransfer	[SSTR]
scheduleSubmitTransferSet	[SSTS]
transferStarted	[TSTR]
transferCancelled	[TCAN]
transferComplete	[TCOM]
transferDelete	[TDEL]
transferProgress	[TPRO]

Excluding message types from the MFT stand-alone file logger

If you want to exclude a certain message type from the file logger output, you can use empty message type elements.

Example

For example, the following format definition stops `transferProgress` messages being output by the file logger.

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="1.00"
  xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferProgress></transferProgress>
  </messageTypes>
</logFormatDefinition>
```

Defining custom formats for the MFT stand-alone file logger

It is possible to define a subset of custom message types within a log format definition to reduce the amount of configuration required to customize your log file format.

About this task

If a `messageTypes` element is not included in the `FileLoggerFormat.xml` file, the format for that message type uses the default format. You need only to specify the formats you want to be different from the default.

Example

In this example, the format definition replaces the default format for the `transferStarted` message type, with this reduced version that outputs only the user that started the transfer. All other message types use the default format because they are not included in this log format definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="1.00"
  xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferStarted>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false">/transaction/action/
            @time</insert>
          <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/originator/
            userID</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferStarted>
  </messageTypes>
</logFormatDefinition>
```

Related reference

[“MFT stand-alone file logger default log format” on page 587](#)

Default log file format definition for the Managed File Transfer stand-alone file logger.

[“Stand-alone file logger format XSD” on page 591](#)

The schema for a stand-alone file format.

Reducing duplicate messages in the MFT stand-alone file logger

Duplicate log messages can occur in the log of the stand-alone file logger. By using the `logger.properties` file you can tune the stand-alone file logger and reduce the number of duplicates.

Duplicate messages in the file logger log

In the case of a failure, a log message might be written to the log of the stand-alone file logger without the consumption of the log message from the `SYSTEM.FTE/Log#` topic being committed to IBM MQ. If this happens, when the stand-alone file logger restarts it will retrieve the same message a second time and write it to the log file again. Plan to handle the possibility of these duplicates when looking at the log files either manually or when processing them automatically. To assist in the detection of duplicates, the stand-alone file logger outputs the following message to the log file when it starts:

```
BFGDB0054I: The file logger has successfully started
```

Duplicates always happen around the start time of the stand-alone file logger, because this is when the last message read before the previous instance failed is processed. By knowing when the new instance has started you can detect if duplicates should be expected, and whether they need to be handled or not.

Reducing the number of duplicates

The stand-alone file logger groups together log messages it processes into transactions to improve performance. This batch size is the maximum number of duplicate messages you may see in the case of a failure. To reduce the number of duplicates you can tune the following property in the `logger.properties` file:

```
wmqfte.max.transaction.messages
```

For example, by setting this to 1 the maximum number of duplicated messages is reduced to 1. Be aware that modifying this value has an effect on the performance of your stand-alone file logger so thorough testing is required to ensure this does not adversely affect your system.

The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information on available properties and their effects, see [“MFT logger configuration properties” on page 111](#)

Configuring user access for an MFT stand-alone file logger

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

About this task

You must install the stand-alone file logger and IBM MQ on a single system. Configure the user's permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of the Managed File Transfer installation.
2. Ensure that the user has permission to create and write to any file in the `logs` directory which is in the configuration directory. This directory is used for an event log, and if necessary for diagnostic trace and First Failure Data Capture (FFDC) files.
3. Ensure that the user has its own group, and is also not in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the `mqm` group. On certain platforms,

the staff group is automatically given queue manager access as well; the stand-alone file logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it by using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands `dspmqaout` (display authority) or `dmpmqaut` (dump authority).

4. Use the **Manage Authority Records** window in the IBM MQ Explorer or the `setmqaut` (grant or revoke authority) command to add authorities for the user's own group (on UNIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
 - Connect and Inquire on the queue manager (the IBM MQ Java libraries require Inquire permission to operate).
 - Subscribe permission on the SYSTEM.FTE topic.
 - Put permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
 - Get permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone file logger queues, add the permissions to those queue names instead.

Installing the MFT stand-alone database logger

Complete these steps to install and configure the stand-alone database logger.

About this task

Important: Managed File Transfer loggers are not supported on the IBM i platform.

For more information about the stand-alone database logger, see [“Configuring an MFT logger” on page 100](#).

Note: You cannot run more than one database logger (stand-alone or Java EE) against the same schema in a database at any one time. Attempting to do so would result in clashes when attempting to write transfer log data to the database.

Procedure

1. Install your database software using the documentation for your database.
If JDBC support is an optional component for your database, you must install this component.
2. Run the **fteCreateLogger** command setting the parameter **-loggerType** to DATABASE to create your stand-alone database logger. For more information, see [“fteCreateLogger \(create an MFT file or database logger\)” on page 389](#).

The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this schema name before proceeding to the next step. For more information, see `wmqfte.database.schema` in [“MFT logger configuration properties” on page 111](#).

3. Create the required database tables using your database's tools.

 On [Multiplatforms](#), the files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables.

 On z/OS, the file that you need to run depends on the version of Db2 for z/OS that you are using:

- For Db2 for z/OS 9.0 and earlier, run the file `ftelog_tables_zos.sql` to create the tables. This file creates the tables using an INTEGER data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.

- For Db2 for z/OS 9.1 and later, run the file `ftelog_tables_zos_bigint.sql` to create the tables. This file creates the tables using a BIGINT data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
4. Run the MQSC commands, provided by the **fteCreateLogger** command, against your logger command queue manager to create the logger queues. The stand-alone database logger uses two queues on the coordination queue manager. The first queue is a command queue where messages to control the operation of the stand-alone database logger are placed. The default name of this command queue is `SYSTEM.FTE.LOG.CMD.logger_name`. The second queue is a reject queue. Because the stand-alone database logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination, and possible reprocessing. You are not recommended to use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages should not be combined with messages put to the dead letter queue for other reasons. The default name for the reject queue is `SYSTEM.FTE.LOG.RJCT.logger_name`. These two queues are defined in the MQSC script files generated by the **fteCreateLogger** command.
 5. [Choose a user and configure permissions](#)
 6. Optional: You can configure the stand-alone database logger further by editing the `logger.properties` file created by the **fteCreateLogger** command in step “2” on [page 108](#). This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information about available properties and their effects, see “[MFT logger configuration properties](#)” on [page 111](#).
 7.  **Windows**
Optional: If you are using a Windows system, you can run the stand-alone database logger as a Windows service. Run the **fteModifyLogger** command with the **-s** parameter. For more information, see “[fteModifyLogger \(run an MFT logger as a Windows service\)](#)” on [page 471](#).
 8. Optional: If the database being used is Oracle or you are connecting to a Db2 database remotely, you will need to specify a user name and password that the logger will use to authenticate with your database server. This user name and password is specified in a credentials file that conforms to the format defined by the `MQMFTCredentials.xsd` schema. For more information, see “[MFT credentials file format](#)” on [page 785](#). After creating the credential file, you must specify the location of the credentials file in the `logger.properties` file using the `wmqfte.database.credentials.file` property.
 9. Start the stand-alone database logger using the **fteStartLogger** command. By default, the stand-alone database logger runs in the background and the stand-alone database logger places output into a file in the `logs` directory. If you want to run the stand-alone database logger in the foreground and produce output to the console as well as to the log file, add the **-F** parameter to the **fteStartLogger** command.

If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone database logger starts as a Windows service.

Using MFT with a remote database

You can use the Managed File Transfer logger to communicate with a database on a remote system.

About this task

If you have a database installed on a different machine from the machine Managed File Transfer is installed on, complete the following steps. The steps apply to both Db2 and Oracle unless stated otherwise.

Procedure

1. Install a database client on the system that you have installed Managed File Transfer on.
2. Add your remote database server to your local database client configuration. This configuration update is needed for Managed File Transfer and IBM MQ to correctly access the database.

3. Specify the new properties in the `logger.properties` file to connect to the database using the credentials file: **`wmfte.database.credentials.file`**.

Note: Earlier versions of Managed File Transfer used the properties **`wmqfte.oracle.user`** or **`wmqfte.database.user`**, and **`wmqfte.oracle.password`** or **`wmqfte.database.password`**. These properties are now deprecated. Use **`wmfte.database.credentials.file`** instead.

4. **Oracle only:** To allow a remote connection to the database, change the XAResourceManager stanza in the coordination queue manager's `qm.ini` file to the following (ensuring you change the database name, user name and user password to match your own information):

```
Oracle_XA+Acc=P/ftelog/  
qgw783jhT+SesTm=35+DB=FTEAUDIT1+SqlNet=FTEAUDIT1+threads=false,  
the change is highlighted in bold.
```

5. **Oracle only:** Specify a host and port in the `logger.properties` file, using the **`wmqfte.oracle.host`** and **`wmqfte.oracle.port`** properties. The default values for the host and port allow you to work with a local database client so if you have previously worked with a local database, you might not have set these values.

Related reference

[“MFT logger configuration properties” on page 111](#)

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

Configuring user access for an MFT stand-alone database logger

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

About this task

The number and type of user accounts you need to run the stand-alone database logger depends on the number of systems you use. You can install the stand-alone database logger, IBM MQ and your database on a single system, or across two systems. The stand-alone database logger must be on the same system as IBM MQ. The components can be installed in the following topologies:

Stand-alone database Logger, IBM MQ and the database all on the same system

You can define a single operating system user for use with all three components. This is a suitable configuration for the stand-alone database logger. The stand-alone database logger uses Bindings mode to connect to IBM MQ and a native connection to connect to the database.

Stand-alone database Logger and IBM MQ on one system, the database on a separate system

You create two users for this configuration: an operating system user on the system running the stand-alone database logger, and a operating system user with remote access to the database on the database server. This is a suitable configuration for the stand-alone database logger using a remote database. The stand-alone database logger uses Bindings mode to connect to IBM MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `ftelog`, but you can use any user name. Configure the user's permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of the Managed File Transfer Remote Tools and Documentation installation.
2. Ensure that the user has permission to create and write to any file in the `logs` directory (in the configuration directory). This directory is used for an event log, and if necessary for diagnostic trace and FFDC files.
3. Ensure that the user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the `mqm` group. On certain platforms, the

staff group is automatically given queue manager access as well; the stand-alone database logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands `dspmqaout` (display authority) or `dmpmqaut` (dump authority).

4. Use the **Manage Authority Records** window in the IBM MQ Explorer or the `setmqaut` (grant or revoke authority) command to add authorities for the user's own group (on UNIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
 - Connect and Inquire on the queue manager (the IBM MQ Java libraries require Inquire permission to operate).
 - Subscribe permission on the SYSTEM.FTE topic.
 - Put permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
 - Get permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone database logger queues, add the permissions to those queue names instead.

5. Perform the user configuration that is specific to the database you are using.
 - If your database is Db2, carry out the following steps:

There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.

 - Ensure that the `fte1og` user is not in any Db2 administration groups (for example, `db2iadm1`, `db2fadm1`, or `dasadm1`)
 - Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#)
 - If your database is Oracle, carry out the following steps:
 - Ensure that the `fte1og` user is not in any Oracle administration groups (for example, `ora_dba` on Windows or `dba` on UNIX)
 - Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#)

MFT logger configuration properties

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

For IBM WebSphere MQ 7.5, or later, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see [“The use of environment variables in MFT properties”](#) on page 511.

Note: When you specify file paths on Windows, the backslash (\) separator character must appear as double backslashes (\\) (that is, escaped \). Alternatively, you can use a single forward slash character (/) as a separator. For more information about character escaping in Java properties files in Oracle, see [Javadoc for the Properties class](#).

Bindings mode connection properties

Property name	Description	Default value
<code>wmqfte.logger.type</code>	The logger type in use: file, or database. Set this value to FILE, or DATABASE.	No default value

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.max.transaction.messages	The maximum number of messages that is processed in a transaction before the transaction is committed. In circular logging mode, a queue manager has a fixed amount of space available for inflight data. Ensure that you set this property with a sufficiently low value so that the available space does not run out.	50
wmqfte.max.transaction.time	The maximum length of time in milliseconds that passes between transaction commits.	5000
wmqfte.max.consecutive.reject	The maximum number of messages that can be rejected consecutively (that is, without encountering a valid message). If this number is exceeded the logger concludes that the problem is not with the messages themselves but with the configuration. For example, if you make an agent-name column in the database narrower than all of your agent names, all messages referring to agents are rejected.	50
wmqfte.reject.queue.name	The name of a queue to which the logger puts messages that the logger cannot handle. If you have a database logger see Database logger error handling and rejection for details of which messages might be put onto this queue.	SYSTEM.FTE.LOG.RJCT. <i>logger_name</i>
wmqfte.command.queue.name	The name of a queue that the logger reads command messages controlling its behavior from.	SYSTEM.FTE.LOG.CMD. <i>logger_name</i>
wmqfte.queue.manager	The queue manager that the logger connects to. This parameter is required, and is all that is needed for bindings mode connections to the queue manager. (For the properties for connecting to a remote queue manager, see Table 17 on page 118.)	No default value
wmqfte.message.source.type	One of the following values: automatic subscription The default value. The logger creates and uses its own durable, managed subscription on the queue manager that is defined in SYSTEM.FTE/Log/#. This is an appropriate value for most scenarios. administrative subscription If the automatic subscription is not appropriate, you can define a different subscription (for example, by using the IBM MQ Explorer, MQSC, or PCF) and instruct the logger to use that subscription. For example, use this value to partition the log space so that one logger handles agents from A-H, another logger handles I-P, and a third logger from Q-Z. queue If the IBM MQ topology means that creating a subscription for the logger is not convenient, you can use a queue instead. Configure IBM MQ so that the queue receives the messages that are typically received by a subscription to SYSTEM.FTE/Log/# on the coordination queue manager.	automatic subscription
wmqfte.message.source.name	If the message source type is administrative subscription or queue, the name of the subscription or queue to use. This property is ignored if the source type is automatic subscription.	No default value

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
<p>wmqfte.database.credentials.file</p>	<p>The file that contains the user name and password for connecting to the database.</p> <p>For IBM WebSphere MQ 7.5, or later, the value of this property can contain environment variables.</p> <p>For more information, see “MFT credentials file format” on page 785.</p>	<p>z/OS For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42</p> <p>ULW See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.</p> <p>ULW Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication</p>
<p>wmqfte.database.driver</p>	<p>The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file. For example, the Type 2 driver for Db2 on AIX® systems requires the file <code>/opt/IBM/db2/V9.5/java/db2jcc.jar</code>. On Windows systems, specify the path separator as a forward slash character (/) for example, <code>C:/Program Files/IBM/SQLLIB/java/db2jcc.jar</code>.</p> <p>z/OS On z/OS, specify the full path of the <code>db2jcc.jar</code> file. For example, <code>wmqfte.database.driver=/db2/db2v10/jdbc/classes/db2jcc.jar</code>.</p> <p>z/OS On z/OS systems, you must reference all of the following JAR files:</p> <ul style="list-style-type: none"> • <code>db2jcc.jar</code> • <code>db2jcc_license_cisuz.jar</code> • <code>db2jcc_javax.jar</code> <p>If your database driver consists of multiple JAR files (for example, Db2 9.1 requires a driver JAR file and a license JAR file), include all of these JAR files in this property. Separate multiple file names by using the classpath separator for your platform, that is, the semicolon character (;) on Windows systems and the colon character (:) on other platforms.</p>	<p>No default value</p>

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.database.exclude_duplicate_metadata	<p>Controls whether entries are stored in the metadata table that contains information that can be found in other tables within the database logger schema. Set this value to <code>true</code>, or <code>false</code>. These metadata entries are no longer stored by default as it is duplication of existing data and a waste of database storage capacity. The property entries and the tables, where the same data appears, are as follows:</p> <ul style="list-style-type: none"> • <code>com.ibm.wmqfte.SourceAgent TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.DestinationAgent TRANSFER_EVENT</code> • <code>com.ibm.wmqfte.MqmdUser TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.OriginatingUser TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.OriginatingHost TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.TransferId TRANSFER</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.JobName TRANSFER</code> or <code>CALL_REQUEST</code> <p>Setting the value of this property to <code>false</code> causes these metadata entries to be stored in the metadata table.</p>	true
wmqfte.database.host	<p>Db2 only:</p> <p>For IBM WebSphere MQ 7.5, or later, the host name of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for <code>wmqfte.database.port</code> must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver.</p> <p>If a value for this property is specified, then a credentials file for this logger (file path defined by the <code>wmqfte.database.credentials.file</code> property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.</p>	No default value
wmqfte.database.name	<p>The name of the database instance (or subsystem when using Db2 for z/OS) that contains the Managed File Transfer log tables.</p>	No default value
wmqfte.database.type	<p>The database management system in use: Db2 or Oracle. Set this value to <code>db2</code> or <code>oracle</code>.</p>	db2
wmqfte.database.port	<p>Db2 only:</p> <p>For IBM WebSphere MQ 7.5, or later, the port number of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for <code>wmqfte.database.host</code> must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver.</p> <p>If a value for this property is specified, then a credentials file for this logger (file path defined by the <code>wmqfte.database.credentials.file</code> property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.</p>	No default value

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.database.schema	<p>Db2 only:</p> <p>The database schema that contains the Managed File Transfer logging tables. In most cases the default value is appropriate, but you might need to specify an alternative value depending on your own site-specific database considerations.</p>	FTELOG
wmqfte.database.native.library.path	<p>The path that contains the native libraries that are needed by your chosen database driver (if any). For example, the Type 2 driver for Db2 on AIX systems requires libraries from /opt/IBM/db2/V9.5/lib32/. As an alternative to this property, you can set the java.library.path system property by using other methods.</p> <p>On Solaris and HP-UX systems, before running the fteStartLogger command, you must also set and export the LD_LIBRARY_PATH environment variable to include the path.</p>	No default value
wmqfte.file.logger.fileDirectory	The directory where the file logger log files are located.	mqft/logs/coordination_dir/loggers/logger_name/logs
wmqfte.file.logger.fileSize	<p>The maximum size that a log file is allowed to grow to. The size value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example, <code>wmqfte.file.logger.fileSize=5MB</code> Specifies a maximum file size of 5MB.</p> <p><code>wmqfte.file.logger.fileSize=2d</code> Specifies a maximum file size of 2 days of data.</p>	10MB
wmqfte.file.logger.fileCount	The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of files never exceeds the value that is specified.	3

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.file.logger.mode	<p>The logger mode in use: circular, or linear. Set this value to CIRCULAR, or LINEAR.</p> <p>CIRCULAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the wmqfte.file.logger.fileSize property. When the maximum size is reached, the file logger starts a new file. The maximum number of files that are written in this mode is controlled by the value that is defined by using the wmqfte.file.logger.fileCount property. When this maximum number of files is reached, the file logger deletes the first file and re-creates it for use as the currently active file. If the value defined in the wmqfte.file.logger.fileSize property is a fixed size byte unit (for example, KB, MB, or GB) then the upper limit on disk space that is used in this mode equals fileSize multiplied by fileCount. If the value defined in the wmqfte.file.logger.fileSize property is a time unit (for example, m, h, d, or w) then the maximum size depends on the throughput of log messages in your system over these time periods. The log file naming convention that is used when running in this mode is: <i>logger_name</i>number-<i>timestamp</i>.log where:</p> <ul style="list-style-type: none"> • <i>logger_name</i> is the name that is given to the logger in the fteCreateLogger command. • <i>number</i> is the number of the file within the set. • <i>timestamp</i> is the timestamp of when the file was created. <p>For example, LOGGER1-20111216123430147.log</p> <p>LINEAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the wmqfte.file.logger.fileSize property. When the maximum size is reached the file logger starts a new file. Previously written files are not deleted, which allows them to be kept as a historical record of log messages. Files are not deleted when running in linear mode, so the wmqfte.file.logger.fileCount property is ignored because there is no upper limit to the number of files that can be created. Because there is no upper limit when running in this mode, it is necessary to track the amount of disk space that is used by the log files to avoid running low on disk space. The log file naming convention that is used when running in this mode is: <i>logger_name</i>-<i>timestamp</i>.log where:</p> <ul style="list-style-type: none"> • <i>logger_name</i> is the name that is given to the logger in the fteCreateLogger command. • <i>timestamp</i> is the timestamp of when the file was created. <p>For example, LOGGER-20111216123430147.log</p>	No default value

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.max.retry.interval	<p>The maximum time, in seconds, between retries when the logger encounters a persistent error.</p> <p>Some error conditions (for example, loss of database connection) prevent the logger from continuing. When this type of condition occurs, the logger rolls back the current transaction, waits for a period, and then retries. The time that the logger waits is initially very short, so that transitory errors can be overcome quickly. However, each time the logger retries, the time that it waits is increased. This prevents too much unnecessary work from taking place when the error condition is longer-lasting, for example when a database is taken down for maintenance.</p> <p>Use this property to set a limit to the length of the wait so that a retry occurs in a reasonable time of the error condition being resolved.</p>	600
loggerQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the logger's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval specified by the value of the maxRestartInterval property. When this value is exceeded the logger's process controller stops restarting the logger, and instead performs an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the logger's process controller measures logger restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the logger's process controller stops restarting the logger. Instead the logger's process controller performs an action that is based on the value of the maxRestartDelay property.	120
maxRestartDelay	Determines the behavior of the logger's process controller when the rate of logger restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the logger's process controller is stopped. If you specify a value greater than zero, this is the number of seconds to wait before the restart history information held by the logger's process controller is reset and the logger is restarted.	-1
wmqfte.oracle.port	The port that the logger uses to connect to the Oracle instance. This port is also known as a TNS listener.	1521
wmqfte.oracle.host	The host that the logger uses to connect to the Oracle instance.	localhost
armELEMTYPE	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMTYPE parameter value specified in the associated ARM policy. For a logger, set ELEMTYPE to SYSBFGLG.	Not set
armELEMENT	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMENT parameter value specified in the associated ARM policy. You can set the ELEMENT value to correspond to the logger name.	Not set

Table 16. Connection properties for bindings mode (continued)

Property name	Description	Default value
loggerQMgrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the logger's coordination queue manager.	 For details on creating the Authentication Credentials File see "Configuring MQMFTCredentials.xml on z/OS" on page 42  See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.  Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication
trace	Optional property. Trace specification when the logger is to be run with trace enabled at logger start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, <code>com.ibm.wmqfte.databaselogger,com.ibm.wmqfte.databaselogger.operation=all</code> . You can specify multiple trace specifications in a colon-separated list. For example, <code>com.ibm.wmqfte.databaselogger=moderate:com.ibm.wmqfte.databaselogger.operation=all</code>	None
traceFiles	Optional property. The total number of trace files to keep. This value applies to the process controller of a logger, as well as the logger itself.	5
traceSize	Optional property. The maximum size in MB of each trace file, before trace wraps onto the next file. This value applies to the process controller of the logger, and the logger itself.	20

Client mode connection properties

V 9.0.4

The following table shows the additional logger properties required to support client mode connection to a logger queue manager.

Table 17. Connection properties for client mode

Property name	Description	Default value
wmqfte.queue.manager.host	Host name, or IP address, of logger queue manager.	No default value
wmqfte.queue.manager.port	Port on which the logger queue manager is listening.	1414
wmqfte.queue.manager.channel	Name of the server connection channel on the logger queue manager.	SYSTEM.DEF.SVRCONN
wmqfte.Ssl.CipherSuite	Specifies TLS aspects of how the logger and the logger queue manager exchange data. The value of <code>wmqfte.Ssl.CipherSuite</code> is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the logger queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.	No default value

Table 17. Connection properties for client mode (continued)

Property name	Description	Default value
wmqfte.Ssl.PeerName	Specifies a distinguished name skeleton that must match the name that is provided by the logger queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	No default value
wmqfte.Ssl.TrustStore	Specifies the location of the certificates that the logger trusts. The value of wmqfte.Ssl.TrustStore is a file path. If the file path is a Windows file path the backslash character (\) must be escaped with a further backslash character (\\). Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.TrustStoreCredentialsFile	The path to the file that contains the wmqfte.Ssl.TrustStore credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.TrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.KeyStore	Specifies the location of the private key of the logger. The value of wmqfte.Ssl.KeyStore is a file path. If the file path is a Windows file path the backslash character (\) must be escaped with a further backslash character (\\). Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStore.CredentialsFile	The path to the file that contains the wmqfte.Ssl.KeyStore credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.FipsRequired	Specifies that you want to enable FIPS support at the level of the logger. The value of this property can be true or false. For more information, see “FIPS support in MFT” on page 641 .	false

Related reference

[“The use of environment variables in MFT properties” on page 511](#)

From IBM WebSphere MQ 7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

Alternative configurations for an MFT stand-alone logger

Typically a Managed File Transfer stand-alone logger, whether it is a file or a database type, is on the same system as the coordination queue manager and is connected to the coordination queue manager in IBM MQ bindings mode. However, it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone logger receives messages using a subscription, which the stand-alone logger creates automatically. This is the configuration described in the installation instructions.

However, if you have site-specific considerations, you can configure a stand-alone logger to receive messages in two other ways, controlled by the `wmqfte.message.source.type` property. This property is described in [Database Logger properties](#).

Administrative subscription

By default, a stand-alone logger creates its own subscription to the `SYSTEM.FTE/Log/#` topic, using the default durable subscription options and a managed subscription (that is, the queue manager controls the backing queue used to hold the messages before they are passed to the application). If other options are required on the subscription or the queue, you can instead create a subscription yourself, set the options that you require, and configure the stand-alone logger to use that subscription instead. Remember to add permission for the stand-alone logger to use the subscription that you create.

An example of using this configuration is to partition the log space by using two wildcard subscriptions, to send logs from agents whose name begins with `FINANCE` into one database and logs from agents beginning with `ACCOUNTING` into another. This type of configuration requires two stand-alone logger instances, each with its own `logger.properties` file referring to the required subscription and its own command queue and reject queue.

To collect log messages only from agents whose names begin with `ACCOUNTING`, create a subscription object on your coordination queue manager with a topic string of `SYSTEM.FTE/Log/ACCOUNTING*`. Set the **Wildcard usage** value to **Character level wildcard**. You must also add entries to the `logger.properties` file for your logger. For example, if you create a subscription object called `ACCOUNTING.LOGS` with these settings, add the following entries to the `logger.properties` file:

```
wmqfte.message.source.type=administrative subscription
wmqfte.message.source.name=ACCOUNTING.LOGS
```

The stand-alone logger handles log messages that start with the topic string of `SYSTEM.FTE/Log/` only. You can specify a more restrictive topic string, but you cannot specify a less restrictive string. If you do specify a less restrictive string in error, all publications that relate to a topic string other than `SYSTEM.FTE/Log/` go to the reject queue, and the stand-alone logger produces the error message `BFGDB0002E`. This error message implies that there is a problem with the stand-alone logger configuration.

Queue

The typical topology is where the stand-alone logger runs on the same system as the coordination queue manager. If this is not possible, you can create a subscription on the coordination queue manager using a queue on another queue manager as the subscription destination (either using a remote queue definition or by using the `DESTQMGR` property of the subscription). The logger can then run on the system hosting the second queue manager and read the messages from the queue. To ensure transactional integrity, the stand-alone logger must always connect to its queue manager in bindings mode. You must define the reject queue and command queue on the same queue manager that the stand-alone logger connects to. The queue managers must be at IBM WebSphere MQ 7.5 or later.

For example, to collect log messages which are being placed on the queue USER.QUEUE by a subscription, add these entries to the `logger.properties` file:

```
wmqfte.message.source.type=queue
wmqfte.message.source.name=USER.QUEUE
```

Installing the Java EE database logger for MFT

Follow these instructions to install and configure the Java EE database logger for use with Managed File Transfer.

About this task

For more information about the Java EE database logger, see the topic [“Configuring an MFT logger” on page 100](#).

Note: You cannot run a Java EE database logger at the same time as a stand-alone logger, unless these loggers are using separate instances of the database.

Procedure

1. Before installing the Java EE database logger, you must prepare your environment. Use the instructions in the topic [“Preparing to install the Java EE database logger for MFT” on page 122](#).
2. You install the Java EE database logger in a Java Platform, Enterprise Edition (Java EE)-compliant application server. For instructions, see the following topics:
 - [“Installing the Java EE database logger for MFT with WebSphere Application Server 7.0” on page 124](#)
 - [“Installing the Java EE database logger for MFT with WebSphere Application Server Community Edition” on page 129](#)

Related tasks

[“Preparing to install the Java EE database logger for MFT” on page 122](#)

Follow these instructions to prepare your Managed File Transfer environment before installing the Java EE database logger.

[“Installing the Java EE database logger for MFT with WebSphere Application Server 7.0” on page 124](#)

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 7.0.

[“Installing the Java EE database logger for MFT with WebSphere Application Server Community Edition” on page 129](#)

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server Community Edition.

[“Configuring user access for the Java EE database logger for MFT” on page 132](#)

When you configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer, you need user accounts to access IBM MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

[“Migrating from the stand-alone database logger to the Java EE database logger for MFT” on page 134](#)

You can migrate from the stand-alone database logger to the Java EE database logger. You must stop the stand-alone database logger and install the Java EE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration.

Related reference

[“Authorities for the MFT logger” on page 345](#)

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

Preparing to install the Java EE database logger for MFT

Follow these instructions to prepare your Managed File Transfer environment before installing the Java EE database logger.

About this task

For more information about the Java EE database logger, see the topic [“Configuring an MFT logger” on page 100](#).

Procedure

1. Install your database software using the documentation for your database.

If JDBC support is an optional component for your database, you must install this component.

2. Create a database using the tools provided by your database. The database must have a tablespace and bufferpool page size of at least 8K.

The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this before proceeding to the next step.

3. Create the required database tables using your database's tools.

Multi On [Multiplatforms](#), the files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables.

z/OS On z/OS, the file that you need to run depends on the version of Db2 for z/OS that you are using:

- For Db2 for z/OS 9.0 and earlier, run the file `ftelog_tables_zos.sql` to create the tables. This file creates the tables using an INTEGER data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
 - For Db2 for z/OS 9.1 and later, run the file `ftelog_tables_zos_bigint.sql` to create the tables. This file creates the tables using a BIGINT data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
4. If you have changed the schema name from FTELOG, you must change the schema name in the EAR file. For more information, see [“Changing the schema name in your Java EE database logger for MFT” on page 123](#).
 5. Create a reject queue in IBM MQ.

Because the logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination and possible reprocessing. Do not use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages must not be combined with messages put to the dead letter queue for other reasons. The **fteCreateLogger** command creates a reject queue. The default name for this reject queue is `SYSTEM.FTE.LOG.RJCT.logger_name`
 6. Follow the instructions in the topic [Configuring user access for the JEE logger](#).

What to do next

Now you can install the Java EE database logger in a Java EE-compliant application server. Use the instructions in the following topics, based on the application server you are using:

- [“Installing the Java EE database logger for MFT with WebSphere Application Server 7.0” on page 124](#)
- [“Installing the Java EE database logger for MFT with WebSphere Application Server Community Edition” on page 129](#)

Changing the schema name in your Java EE database logger for MFT

The Java Platform, Enterprise Edition (Java EE) database logger can use a database that has a non-default schema name. You must change the schema name in the Java EE database logger EAR file.

About this task

To change the name of the schema that your Java EE database logger uses, complete the following steps:

Procedure

1. Extract the JPA JAR file from the EAR file by using the following command:

```
jar -xvf ear_file lib/jpa_file
```

where:

- *ear_file* is `com.ibm.wmqfte.databaseslogger.jee.oracle.ear` or `com.ibm.wmqfte.databaseslogger.jee.ear` depending on whether you are using Db2 or Oracle.
- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

2. Extract the `persistence.xml` file from the JPA JAR file by using the following command:

```
jar -xvf lib/jpa_file META_INF/persistence.xml
```

where:

- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

3. Edit the `persistence.xml` file to change the following line:

```
<property name="openjpa.jdbc.Schema" value="schema_name"/>
```

where

- *schema_name* is the schema name you want to use.

4. Update JPA JAR with the modified `persistence.xml` file by using the following command:

```
jar -uvf lib/jpa_file META_INF/persistence.xml
```

where:

- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

5. Update the EAR file with the modified JPA JAR file by using the following command:

```
jar -uvf ear_file lib/jpa_file
```

where:

- *ear_file* is `com.ibm.wmqfte.databaseslogger.jee.oracle.ear` or `com.ibm.wmqfte.databaseslogger.jee.ear` depending on whether you are using Db2 or Oracle.
- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

What to do next

Use the modified EAR file to install the Java EE database logger.

Related tasks

[“Installing the Java EE database logger for MFT with WebSphere Application Server 7.0” on page 124](#)
Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 7.0.

[“Installing the Java EE database logger for MFT with WebSphere Application Server Community Edition” on page 129](#)

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server Community Edition.

Setting the native library path in WebSphere Application Server 7.0

If you deploy the Java Platform, Enterprise Edition (Java EE) database logger application on WebSphere Application Server 7.0, and you want to use bindings mode connections between the application and IBM MQ, you must configure the IBM MQ messaging provider with the location of the IBM MQ native libraries on the system.

About this task

If you do not set the native library path in your application server, you might receive the following error message in the WebSphere Application Server 7.0 system out log:

```
A connection could not be made to WebSphere MQ for the following reason:  
CC=2;RC=2495;AMQ8568: The native JNI library 'mqjbd' was not found. [3=mqjbd]
```

Use the WebSphere Application Server 7.0 administrative console to complete the following steps:

Procedure

1. In the navigation pane, expand **Resources > JMS > JMS Providers**.
2. Select the IBM MQ messaging provider that is at the correct scope for the connection factory or activation specification that creates the bindings mode connection.
Note: Native path information at Server scope is used in preference to native path information at higher scopes, and native path information at Node scope is used in preference to native path information at Cell scope.
3. Under General Properties, in the **Native library path** field, enter the full name of the directory that contains the IBM MQ native libraries.
For example, on Linux enter `/opt/mqm/java/lib`. Enter only one directory name.
4. Click **OK**.
5. Restart the application server to refresh the configuration.
6. Required: Restart the application server a second time to load the libraries.

Installing the Java EE database logger for MFT with WebSphere Application Server 7.0

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 7.0.

Before you begin

Before you install the JEE database logger application, follow the instructions in the topics [“Preparing to install the Java EE database logger for MFT” on page 122](#) and [“Setting the native library path in WebSphere Application Server 7.0” on page 124](#).

About this task

For more information about the Java EE database logger, see [“Configuring an MFT logger” on page 100](#).

Procedure

1. Set up the XA JDBC provider:
 - a) Select **Resources > JDBC > JDBC Providers** from the WebSphere Application Server 7.0 administration console navigation.
 - b) Create a JDBC provider using the console wizard, by clicking **New**.
 - c) At Step 1 of the wizard, select the database that you are using from the **Database type** list, and the associated provider type from the **Provider type** list. From the **Implementation type** list, select **XA data source**. Click **Next**.
 - d) At Step 2 of the wizard, ensure that the directory location of the required database jar files is set correctly. Click **Next**.
 - e) Click **Finish** on the summary page to create the JDBC provider.
2. Create authentication aliases. You create one alias for the data source and another for IBM MQ:
 - a) Select **Security > Global security** from the WebSphere Application Server 7.0 administration console navigation.
 - b) Under the **Authentication** heading, expand **Java Authentication and Authorization Service**.
 - c) Click **J2C authentication data**. The authentication alias page opens.
 - d) Create an authentication alias for your data source:
 - i) Click **New**.
 - ii) Enter the details for **Alias, User ID, Password, and Description**. The details that are entered in the **User ID** and **Password** fields must match the details that you entered when you created your database user. For more information, see [Configuring user access for the JEE database logger](#).
 - iii) Click **OK**.
 - e) Create an authentication alias for IBM MQ:
 - i) Click **New**.
 - ii) Enter the details for **Alias, User ID, Password, and Description**. The details that are entered in the **User ID** and **Password** fields must match your user and password settings for your IBM MQ installation.
 - iii) Click **OK**.
3. Create a data source:
 - a) Select **Resources > JDBC > Data sources** from the WebSphere Application Server 7.0 administration console navigation.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.
 - c) Create a data source using the console wizard, by clicking **New**.
 - d) At Step 1 of the wizard, in the **Data source name** field, enter `wmqfte-database` and in the **JNDI name** field, enter `jdbc/wmqfte-database`. Click **Next**.
 - e) At Step 2 of the wizard, use the **Select an existing JDBC provider** dropdown list to select the JDBC provider created in the previous steps. Click **Next**.
 - f) **Db2:** At Step 3 of the wizard, in the **Driver type** field, enter 4.
 - g) **Db2:** Enter the details in the **Database name, Server name, and Port number** fields, and click **Next**.

Oracle: Enter the connection URL in the **URL** field and choose the correct data store helper in the **Data store helper class name** field.

Oracle RAC: When connecting to an Oracle Real Application Cluster, the connection URL must include the host information necessary to connect to all available instances of the database.

- h) At Step 4 of the wizard, select the name of the data source authentication alias that you defined in step 2d from the **Authentication alias for XA recovery** list. Select the same name from the **Component-managed authentication alias** and **Container-managed authentication alias** lists.
 - i) Click **Finish** on the summary page to create the data source.
4. Optional: Verify the configuration of the data source:
- a) Select **Resources > JDBC > Data sources** from the WebSphere Application Server 7.0 administration console navigation.
 - b) Click the **Test Connection** button.
5. Create a topic.
- a) From the WebSphere Application Server 7.0 administration console navigation, click **Resources > JMS > Topics**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) On the **Administration** panel of the property page for the topic, choose unique values for the **Name** and **JNDI name** fields, that you will reference later on in the configuration.
 - f) In the **IBM MQ topic** panel, enter SYSTEM.FTE/Log/# in the **Topic name** field.
6. Create an activation specification:
- a) From the WebSphere Application Server 7.0 administration console navigation, click **Resources > JMS > Activation specifications**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.
 - f) In Step 1.1, enter the JNDI name for the topic that you set up in step 5 in the **Destination JNDI name** field.
 - g) From the **Destination type** list, select **Topic**.
 - h) In Step 1.2 of the wizard, select **Durable Subscription**. Enter SYSTEM.FTE.DATABASELOGGER.AUTO in the **Subscription name** field.
 - i) In Step 2 of the wizard, select **Enter all the required information into this wizard**.
 - j) In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.
 - k) In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.
 - l) Optional: Click **Test Connection** to confirm the queue manager is present. However, you can expect to receive NOT_AUTHORIZED until you have referenced the authentication alias in step 6n.
 - m) Click **Save**.
 - n) Click the name of the Activation Specification that you created. In the **General Properties** section of the **Configuration** tab, scroll down to the **Advanced** panel and enter a unique name to identify your IBM MQ connection in the **Client ID** field. You must complete this step or your connection is rejected by IBM MQ with the JMSCC0101 error code.
 - o) If you chose **Client** as your transport method, scroll down to the **Security Settings** panel and select the authentication alias that you defined in step 8 from the **Authentication alias** list.
 - p) Click **Apply**.

- q) In the **Additional Properties** section of the **Configuration** tab, click **Advanced Properties**. In the **Connection Consumer** section of the **Advanced Properties** panel, enter 1 into the **Maximum server sessions** field.
- Note:** Ensure you complete this step before proceeding. Failure to do so can cause the logger to fail to operate correctly.
- r) In the **Additional Properties** section of the **Configuration** tab, click **Advanced Properties**. Set the value of **Stop endpoint if message delivery fails** to a minimum of 1.
- If the value of the `_numberOfFailedAttemptsBeforeReject` property is set to more than 1 (see 9j for more information), set **Stop endpoint if message delivery fails** to at least the value of the `_numberOfFailedAttemptsBeforeReject` property. This prevents the endpoint from stopping when a message that cannot be processed (for example, a malformed transfer log message) is received. For more information, see [“MFT logger error handling and rejection” on page 315](#).
7. Create a queue connection factory.
- From the WebSphere Application Server 7.0 administration console navigation, click **Resources > JMS > Queue connection factories**.
 - Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.
 - Click **New**.
 - Click **IBM MQ messaging provider**.
 - In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.
 - In Step 2, select **Enter all the required information into this wizard**.
 - In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.
 - In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.
 - Optional: Click **Test Connection** to confirm the queue manager is present. However, you can expect to receive NOT_AUTHORIZED until you have referenced the authentication alias in step 7h.
 - If you selected **Client** or **Bindings then client** as your transport method, click the name of the queue connection factory you have just created. Scroll down to the **Security Settings** panel of the **Configuration** tab and select the authentication alias that you defined in step 2e from the **Authentication alias for XA recovery** and **Container-managed authentication alias** lists.
8. Create a reject queue in WebSphere Application Server:
- From the WebSphere Application Server 7.0 administration console navigation, click **Resources > JMS > Queues**.
 - Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.
 - Click **New**.
 - Click **IBM MQ messaging provider**.
 - Choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.
 - Enter `SYSTEM.FTE.LOG.RJCT.logger_name` in the **Queue name** field. Ensure you have created this queue on your coordination queue manager.
 - Enter your queue manager name in the **Queue manager name** field.
 - Click **OK**.
9. Install the JEE database logger application:
- From the WebSphere Application Server 7.0 administration console, select **Applications > New Application**.

- b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) From the options list, select **New Enterprise Application**.
 - d) On the **Preparing for the application installation** page, select the com.ibm.wmqfte.databaselogger.jee.ear file or the com.ibm.wmqfte.databaselogger.jee.oracle.ear file from the MQ_INSTALLATION_PATH/mqft/web directory of the Managed File Transfer Service installation, and click **Next**.
 - e) On the following screen, select **Detailed** to show all installation options and parameters, and click **Next**.
 - f) Click **Next** through wizard steps 1-4 to accept the default values.
 - g) In step 5 of the wizard, **Bind listeners for message driven beans**, scroll to the **Listener Bindings** section. Click **Activation Specification**.
Enter the required values for the following fields:
 - Target Resource JNDI name**
The JNDI name that you specified when creating an activation specification in step 6d.
 - Destination JNDI name**
The JNDI name that you specified when creating a topic in step 5d.
 Click **Next**.
 - h) In step 6 of the wizard, **Map resource references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name that you specified for the reject queue connection factory in step 7c. Click **Next**.
 - i) In step 7 of the wizard, **Map resource environment entry references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name of the reject queue that you created in step 8d. Click **Next**.
 - j) In step 8 of the wizard, **Map environment entries for EJB modules**, accept the default value of 1. Click **Next**.

Oracle RAC: When connecting to an Oracle Real Application Cluster you must set the value of the _numberOfFailedAttemptsBeforeReject property to **at least 2**. This property determines the number of times that the logger attempts to process an audit message after a failure occurs. In a case of database failover at least one failure is likely to occur. To avoid unnecessarily moving a message to the reject queue, increasing this value allows a second attempt to be made, which usually results in success as a connection is made to the new database instance. If you find during testing that messages are still moved to the reject queue during failover of your database instance, increase this value further: the timing of the switch between instances might cause more than one failure for the same message. However, be aware that increasing this value affects all failure cases (for example, a malformed message) and not just database failover, so increase the value with care to avoid unnecessary retries.
 - k) In step 9 of the wizard, **Metadata for modules**, click **Next**.
 - l) In step 10 of the wizard, **Summary**, click **Finish**.
10. You can now start the application from the WebSphere Application Server 7.0 administration console:
- a) Select **Applications > Application Types > WebSphere enterprise applications** from the console navigation.
 - b) Select the check box for the **Logger** enterprise application from the collection table, and click **Start**.

Installing the Java EE database logger for MFT with WebSphere Application Server Community Edition

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server Community Edition.

Before you begin

Before installing the Java EE database logger application, follow the instructions in the topic [“Preparing to install the Java EE database logger for MFT”](#) on page 122.

About this task

For more information about the Java EE database logger, see the topic [“Configuring an MFT logger”](#) on page 100.

Procedure

1. Deploy the IBM MQ resource adapter, `wmq.jmsra.rar`.
 - To deploy the IBM MQ resource adapter for a JEE database logger using a coordination queue manager QM_JUPITER, perform the following steps. This example applies when your WebSphere Application Server Community Edition instance is running on the same system as the IBM MQ queue manager that you want to connect to.
 - a. Create a plan file that defines a connection to the MFT coordination queue manager. The following example plan file defines a connection to a queue manager called QM_JUPITER, and a reference to a queue called SYSTEM.FTE.LOG.RJCT.LOGGER1 on that queue manager.

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-interface>
        <connectiondefinition-instance>
          <name>jms/WMQFTEJEEEDBLoggerRejectQueueCF</name>
          <config-property-setting name="queueManager">QM_JUPITER</config-property-setting>
          <config-property-setting name="transportType">BINDINGS</config-property-setting>
          <connectionmanager>
            <xa-transaction>
              <transaction-caching/>
            </xa-transaction>
            <single-pool>
              <max-size>10</max-size>
              <min-size>1</min-size>
              <blocking-timeout-milliseconds>5000</blocking-timeout-milliseconds>
              <idle-timeout-minutes>2</idle-timeout-minutes>
              <match-all />
            </single-pool>
          </connectionmanager>
        </connectiondefinition-instance>
      </connection-definition>
    </outbound-resourceadapter>
  </resourceadapter>
  <adminobject>
    <adminobject-interface>javax.jms.Queue</adminobject-interface>
    <adminobject-class>com.ibm.mq.connector.outbound.MQQueueProxy</adminobject-class>
    <adminobject-instance>
      <message-destination-name>jms/WMQFTEJEEEDBLoggerRejectQueue</message-destination-name>
      <config-property-setting name="baseQueueManagerName">QM_JUPITER</config-property-setting>
      <config-property-setting name="baseQueueName">SYSTEM.FTE.LOG.RJCT.LOGGER1</config-property-setting>
    </adminobject-instance>
  </adminobject>
</connector>
```

To use this plan file in your environment change QM_JUPITER to the name of your coordination queue manager.

- b. Open the WebSphere Application Server CE administration console.
 - c. From the **Common Console Actions** list on the **Welcome page**, click **Deploy New Applications > Deploy New**.
 - d. In the **Archive** field, enter `mq_install_root/java/lib/jca/wmq.jmsra.rar`
 - e. In the **Plan** field, type the path to the plan file you created in Step 1a.
- If your WebSphere Application Server Community Edition instance is running on a different system to the IBM MQ queue manager that you want to connect to, perform the following steps to deploy the IBM MQ resource adapter.
 - a. Create a plan file that defines a connection to the WMQFTE coordination queue manager.

The following example plan file defines a connection to a queue manager, QM_SATURN, that is located on a different system to your WebSphere Application Server Community Edition installation, and a reference to a queue called SYSTEM.FTE.LOG.RJCT.LOGGER1 on that queue manager. The host name of QM_SATURN is saturn.example.com. The port of QM_SATURN is 1415. The channel of QM_SATURN is SYSTEM.DEF.SVRCONN.

Because the application server and the queue manager are on different systems, you must use a client mode connection to the queue manager. The following plan file sets the value of the `<config-property-setting>` element that has the name `transportType` to `CLIENT`.

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
  </resourceadapter>
  <outbound-resourceadapter>
    <connection-definition>
      <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-interface>
      <connectiondefinition-instance>
        <name>jms/WMQFTEJEEEDBLoggerRejectQueueCF</name>
        <config-property-setting name="queueManager">QM_SATURN</config-property-setting>
        <config-property-setting name="transportType">CLIENT</config-property-setting>
        <config-property-setting name="channel">SYSTEM.DEF.SVRCONN</config-property-setting>
        <config-property-setting name="hostName">saturn.example.com</config-property-setting>
        <config-property-setting name="port">1415</config-property-setting>
        <connectionmanager>
          <xa-transaction>
            <transaction-caching/>
          </xa-transaction>
          <single-pool>
            <max-size>10</max-size>
            <min-size>1</min-size>
            <blocking-timeout-milliseconds>5000</blocking-timeout-milliseconds>
            <idle-timeout-minutes>2</idle-timeout-minutes>
            <match-all />
          </single-pool>
        </connectionmanager>
      </connectiondefinition-instance>
    </connection-definition>
  </outbound-resourceadapter>
</resourceadapter>
<adminobject>
  <adminobject-interface>javax.jms.Queue</adminobject-interface>
  <adminobject-class>com.ibm.mq.connector.outbound.MQQueueProxy</adminobject-class>
  <adminobject-instance>
    <message-destination-name>jms/WMQFTEJEEEDBLoggerRejectQueue</message-destination-name>
    <config-property-setting name="baseQueueManagerName">QM_SATURN</config-property-setting>
    <config-property-setting name="baseQueueName">SYSTEM.FTE.LOG.RJCT.LOGGER1</config-property-setting>
  </adminobject-instance>
</adminobject>
</connector>
```

To use this plan file in your environment change QM_SATURN to the name of your coordination queue manager. Change the value of the host name, port and channel to the values for your coordination queue manager.

- b. Copy the file `mq_install_root/java/lib/jca/wmq.jmsra.rar` from the system where IBM MQ is installed to the system where WebSphere Application Server CE is installed.
 - c. Open the WebSphere Application Server CE administration console.
 - d. From the **Common Console Actions** list on the **Welcome page**, click **Deploy New Applications > Deploy New**.
 - e. In the **Archive** field, type the path to the copy of the `wmq.jmsra.rar` file that you obtained.
 - f. In the **Plan** field, type the path to the plan file you created.
2. You must define a database connector so that the JEE database logger application has access to the required database from within the WebSphere Application Server Community Edition environment. Carry out the following steps from the WebSphere Application Server Community Edition administration console:
- a) Depending on the level of WebSphere Application Server Community Edition that you are using, from the **Console Navigation** either select **Services > Database Pools**, or select **Resources > Datasources**.
 - b) Create a database pool using the Geronimo database pool wizard. In the **Name of Database Pool** field, type `jdbc/wmqfte-database`.
 - c) For the **Database Type** select `DB2 XA` or `Oracle Thin`, as appropriate for your database.
 - d) Click **Next**.
 - e) In the **Driver jar** field, select the appropriate jar for your database.
 - f) In the **Database Name** field, type the name of the database you are connecting to for transfer status information.
 - g) In the **User Name** field, type the user name for connecting to and authenticating with your database.
 - h) In the **Password** and **Confirm Password** fields, type the password for authenticating with your database.
 - i) In the **Port Number** field, type the port number you are using if it is not the default port.
 - j) Ensure that the value for **Driver Type** is `4`.
 - k) Select `XA` from the **Transaction Type** list.
 - l) Click **Deploy**.
3. Update the Managed File Transfer JEE database logger application `openejb-jar.xml` file for your environment. Use a Java SDK jar utility to complete the following steps:
- a) Extract the EJB jar file from the supplied EAR file by running the following command:

```
jar -xf ear_file_name com.ibm.wmqfte.databaselogger.jee.ejb.jar
```

where `ear_file_name` is `com.ibm.wmqfte.databaselogger.jee.ear` or `com.ibm.wmqfte.databaselogger.jee.oracle.ear` depending on whether you are using Db2 or Oracle. The EAR file is located in the `MQ_INSTALLATION_PATH/mqft/web` directory of the IBM WebSphere MQ File Transfer Edition Server installation.

- b) Extract the `META-INF/openejb-jar.xml` file from the previously extracted EJB jar file, `com.ibm.wmqfte.databaselogger.jee.ejb.jar`, by running the following command:

```
jar -xf com.ibm.wmqfte.databaselogger.jee.ejb.jar META-INF/openejb-jar.xml
```

- c) Use a text editor to edit the extracted `META-INF/openejb-jar.xml` file. Change the following `activation-config-property` values to match your environment:

queueManager

The name of the IBM MQ queue manager that is used by the JEE database logger.

hostName

The host name to use to connect to the specified IBM MQ queue manager. This value is not required if you are connecting to the queue manager in bindings mode.

transportType

Whether to connect to the specified IBM MQ queue manager in client or bindings mode.

port

Not required if you specified a **transportType** of bindings. The port to use to connect to the specified IBM MQ queue manager.

channel

Not required if you specified a **transportType** of bindings. The server channel to use to connect to the specified IBM MQ queue manager.

- d) Update the EJB jar file with the modified META-INF/openejb-jar.xml file, by running the following command:

```
jar -uf com.ibm.wmqfte.databaselogger.jee.ejb.jar META-INF/openejb-jar.xml
```

- e) Update the supplied ear file with the updated EJB jar file, by running the following command:

```
jar -uf ear_file_name com.ibm.wmqfte.databaselogger.jee.ejb.jar
```

where *ear_file_name* is `com.ibm.wmqfte.databaselogger.jee.ear` or `com.ibm.wmqfte.databaselogger.jee.oracle.ear` depending on your database.

4. To deploy the EAR file to the application server, complete the following steps from the WebSphere Application Server Community Edition administration console.

- Select: **Applications > Deploy New** from the **Console Navigation** menu.
- In the **Archive** field, specify the EAR file: `com.ibm.wmqfte.databaselogger.jee.ear` or `com.ibm.wmqfte.databaselogger.jee.oracle.ear` depending on your database.
- Leave the **Plan** field blank.
- Ensure the **Start application after install** box is selected.
- Click **Install**. The JEE database logger application is installed and started.

Configuring user access for the Java EE database logger for MFT

When you configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer, you need user accounts to access IBM MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

About this task

The number and type of user accounts you need to run the Java EE database logger depend on the number of systems you use. User accounts are required to access the following three environments:

- Local operating system
- IBM MQ
- Database

You can install the Java EE database logger, IBM MQ and your database on a single system, or across several systems. The components can be installed in the following example topologies:

Java EE database logger, IBM MQ, and the database all on the same system

You can define a single operating system user for use with all three components. The logger uses Bindings mode to connect to IBM MQ and a native connection to connect to the database.

Java EE database logger and IBM MQ on one system, the database on a separate system

You create two users for this configuration: an operating system user on the system running the logger, and an operating system user with remote access to the database on the database server. The logger uses Bindings mode to connect to IBM MQ and a client connection to access the database.

Java EE database logger on one system, IBM MQ on another system, the database on a further system

You create three users for this configuration: An operating system user to start the application server, an IBM MQ user to access the queues and topics being used, and a database server user to access and insert into the database tables. The logger uses Client mode to access IBM MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `fte1log`, but you can use any user name, new or existing. Configure the user permissions as follows:

Procedure

1. Ensure that the operating system user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the `mqm` group. On certain platforms, the `staff` group is automatically given queue manager access as well; the logger user should not be in the `staff` group. You can view authority records for the queue manager itself and for objects in it using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands `dspmqaout` (display authority) or `dmpmqaout` (dump authority).

2. Use the **Manage Authority Records** window in the IBM MQ Explorer or the `setmqaut` (grant or revoke authority) command to add authorities for the IBM MQ user's own group (on UNIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:

- CONNECT and INQUIRE on the queue manager (the IBM MQ Java libraries require INQUIRE permission to operate).
- SUBSCRIBE permission on the `SYSTEM.FTE` topic.
- PUT permission on the `SYSTEM.FTE.LOG.RJCT.logger_name` queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the logger queues, add the permissions to those queue names instead.

3. Perform the database user configuration that is specific to the database you are using.

- If your database is Db2, carry out the following steps:

Note: There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.

- Ensure that the `fte1log` user is not in any Db2 administration groups (for example, `db2iadm1`, `db2fadm1`, or `dasadm1`)
- Give the user permission to connect to the database and permission to select, insert, and update on the tables that you created as part of [Step 2: create the required database tables](#)

- If your database is Oracle, carry out the following steps:

- Ensure that the `fte1log` user is not in any Oracle administration groups (for example, `ora_dba` on Windows or `dba` on UNIX)
- Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#)

Migrating a Java EE database logger

To migrate a Java EE database logger on WebSphere Application Server 7.0 from IBM WebSphere MQ File Transfer Edition 7.0 to IBM WebSphere MQ 7.5 or later, complete the following steps:

Procedure

1. Open the WebSphere Application Server console.

2. Click **Applications > Application Types > Enterprise Applications**. Locate the IBM WebSphere MQ File Transfer Edition database logger application in the list of applications. If the database logger application is not already stopped, select the application and click **Stop**.
3. Make a note of the configuration settings that you previously set up for the JEE database logger. You will need these later in step “7” on page 134.
 - a) If you originally made changes from the default settings for EJB modules while installing the database logger (see step 9 for more information), click **Enterprise Applications > WebSphere MQ File Transfer Edition database logger > Environment entries for EJB modules** and make a note of the settings in the pane.
 - b) Click **Enterprise Applications > WebSphere MQ File Transfer Edition database logger > Message Driven Bean listener bindings** and make a note of the activation specification used, the **Target Resource JNDI name** and the **Destination JNDI name**.
 - c) Click **Enterprise Applications > WebSphere MQ File Transfer Edition database logger > Resource references** and make a note of the reject queue connection factory details.
 - d) Click **Enterprise Applications > WebSphere MQ File Transfer Edition database logger > Resource environment entry references** and make a note of the reject queue details.
4. Uninstall the IBM WebSphere MQ File Transfer Edition database logger application by clicking **Applications > Application Types > Enterprise Applications**. Select the database logger application and click **Uninstall**.
5. Optional: if you are using multiple installations to migrate to IBM WebSphere MQ 7.5, or later, and the native library path is different, change the path by clicking **Resources > JMS providers > WebSphere MQ messaging provider**

For example, if the native library path was: C:\Program Files\IBM\WebSphere MQ\java\lib, change the path to: C:\Program Files\IBM\New MQ Installation Location\java\lib
6. Optional: if you are using multiple installations to migrate to IBM WebSphere MQ 7.5 or later, you must associate the queue manager with the new installation using the setmqm command.
7. Reinstall the database logger application, using the information in [Installing the IBM MQ Managed File Transfer Java EE database logger with WebSphere Application Server 7.0](#) and the information you recorded earlier in step “3” on page 134.
8. Start the new database logger by clicking **Applications > Application Types > Enterprise Applications**. Select the database logger application and click **Start**.
9. To verify the migration, check the database to ensure that entries are being written.

Migrating from the stand-alone database logger to the Java EE database logger for MFT

You can migrate from the stand-alone database logger to the Java EE database logger. You must stop the stand-alone database logger and install the Java EE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration.

About this task

Procedure

1. Before stopping the database, run the following MQSC command against your coordination queue manager: `ALTER QM PSMODE(COMPAT)`

This stops messages being published to the SYSTEM.FTE/Log topic. Wait until the logger has processed all of the messages on its subscription. By default, this subscription is called SYSTEM.FTE.LOGGER.AUTO.
2. Stop the database logger using the **fteStopLogger** command.
3. Back up the database using the tools supplied with the database software.

4. Delete the subscription belonging to the stand-alone database logger.
By default, this subscription is called SYSTEM.FTE.LOGGER.AUTO.
5. If your database schema is at an earlier version, you must migrate the schema to each subsequent level in order. For example, if your database schema is at 7.0.1 and you are migrating to 7.0.4, you must migrate your schema from 7.0.1 to 7.0.2, then from 7.0.2 to 7.0.3, and then from 7.0.3 to 7.0.4. Migrate your database schema from version *old* to version *new*, where *old* and *new* are variables that describe a schema version, by performing the one of the following actions for each version of the schema that you must migrate through:
 -  If your database is Db2 on z/OS and you are migrating between the 7.0.2 and 7.0.3 schemas or between the 7.0.3 and 7.0.4 schemas, you must create a new database schema and copy your existing data into it. For more information, see [“Migrating the database tables on Db2 on z/OS to IBM MQ 8.0”](#) on page 21.
 - If your database is not Db2 or you created your database with a page size of more than 8K, you can migrate the schema in the same way as for other versions, by completing the following steps.
 - If you are migrating between database tables in any other circumstances complete the following steps:
 - a. Choose the file that is appropriate to your database platform and has a name that includes the string *old-new*. This file is located in the `MQ_INSTALLATION_PATH/mqft/sql` directory of the Remote Tools and Documentation installation.
 - b. If you have made modifications to the initial schema, review the migration file to ensure that the file will be compatible with your modified database.
 - c. Run the SQL file against your database.
6. Install the Java EE database logger EAR file.
7. Deploy the Java EE database logger. For more information, see [“Installing the Java EE database logger for MFT”](#) on page 121.
8. Run the following MQSC command against your coordination queue manager: `ALTER QMGR PSMODE(ENABLED)`
This enables publishing of messages to the SYSTEM.FTE/Log topic.

Results

Configuring the Connect:Direct bridge

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

About this task

Complete the following steps to configure the Connect:Direct bridge:

Procedure

1. [“Choose the operating systems for the Connect:Direct bridge agent and node”](#) on page 136.
2. [“Choose and configure a Connect:Direct node”](#) on page 136.
3. [“Create and configure a Connect:Direct bridge agent”](#) on page 136.
4. [“Configure the ConnectDirectNodeProperties.xml file to include information about the remote Connect:Direct nodes”](#) on page 137.
5. [“Configure a secure connection between the Connect:Direct bridge agent and the Connect:Direct node”](#) on page 138.

Choose the operating systems for the Connect:Direct bridge agent and node

Before you begin

The agent and node that make up the Connect:Direct bridge must be on the same system, or have access to the same file system, for example through a shared NFS mount. This file system is used to temporarily store files during file transfers that involve the Connect:Direct bridge, in a directory defined by the **cdTmpDir** parameter. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to address this directory using the same path name. For example, if the agent and node are on separate Windows systems, the systems must use the same drive letter to mount the shared file system. The following configurations allow the agent and the node to use the same path name:

- The agent and node are on the same system, which is either running Windows or Linux for System x
- The agent is on Linux for System x, and the node is on UNIX
- The agent is on one Windows system, and the node is on another Windows system

The following configurations do not allow the agent and the node to use the same path name:

- The agent is on Linux for System x, and the node is on Windows
- The agent is on Windows, and the node is on UNIX

Consider this restriction when planning your installation of the Connect:Direct bridge.

For more details of the operating system versions supported for the Connect:Direct bridge, see the web page [IBM MQ System Requirements](#).

Procedure

1. Choose a system running either Windows or Linux on System x to install the Connect:Direct bridge agent on.
2. Choose an operating system that is supported by Connect:Direct for Windows or Connect:Direct for UNIX to install the Connect:Direct bridge node on.

Choose and configure a Connect:Direct node

Before you begin

You must have a Connect:Direct node installed before following these instructions.

Procedure

1. Choose a Connect:Direct node for the Managed File Transfer agent to communicate with.
2. Check the network map for your chosen Connect:Direct node. If the network map contains any entries for remote nodes running on a Windows operating system, you must ensure that these entries specify that the nodes are running on Windows.
 - a) If the Connect:Direct node that you have selected for the Connect:Direct bridge is running on Windows, use the Connect:Direct Requester to edit the network map. Ensure that the **Operating System** field for any remote nodes that are running on Windows is set to **Windows**.

Create and configure a Connect:Direct bridge agent

About this task

A Connect:Direct bridge agent is a Managed File Transfer agent that is dedicated to communicating with a Connect:Direct node.

Procedure

1. Create a Connect:Direct bridge agent using the **fteCreateCDAgent** command.

- a) You must provide a value for the **cdNode** parameter. This parameter specifies the name that the agent uses for the Connect:Direct node that is part of the Connect:Direct bridge. Use the name of the Connect:Direct node that you chose in the previous section.
 - b) Provide values for the **cdNodeHost** and **cdNodePort** parameters, which define the Connect:Direct node that the agent communicates with.
If you do not provide a value for the **cdNodeHost** parameter, the host name or IP address of the local system is used. If you do not provide a value for the **cdNodePort** parameter, the value 1363 is used.
 - c) Use the information in [“fteCreateCDAgent \(create a Connect:Direct bridge agent\)”](#) on page 384 to determine whether you need to specify a value for the **cdTmpDir** parameter.
2. Map the user credentials used by Managed File Transfer to user credentials on a Connect:Direct node. You can map credentials by using one of the following methods:
- Create a `ConnectDirectCredentials.xml` file to define credential mapping information. For more information, see [“Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file”](#) on page 138.
 - Write a user exit to perform credential mapping for your Connect:Direct bridge. For more information, see [“Mapping credentials for Connect:Direct by using exit classes”](#) on page 140.

Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes

Before you begin

You must have created a Connect:Direct bridge agent before following these instructions.

Procedure

Edit the template `ConnectDirectNodeProperties.xml` in the Connect:Direct bridge agent configuration directory. For each Connect:Direct node or group of nodes that you want to define information about, perform the following steps:

- a) Inside the `nodeProperties` element, create a `node` element.
- b) Add a `name` attribute to the `node` element. Specify the value of this attribute as a pattern to match the name of one or more remote Connect:Direct nodes.
- c) Optional: Add a `pattern` attribute to the `node` element that specifies what type of pattern the value in the `name` attribute is. Valid values are `regex` and `wildcard`. The default option is `wildcard`.
- d) Add a `type` attribute to the `node` element that specifies the operating system that the remote Connect:Direct nodes specified by the `name` attribute run on.

The following values are valid:

- Windows - the node runs on Windows
- UNIX - the node runs on UNIX or Linux
-  z/OS, zos, os/390, or os390 - the node runs on z/OS

The value of this attribute is not case sensitive. Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

For more information, see [“Connect:Direct node properties file format”](#) on page 563.

Configure a secure connection between the Connect:Direct bridge agent and the Connect:Direct node

About this task

By default, the Connect:Direct bridge agent uses the TCP/IP protocol to connect to the Connect:Direct node. If you want a secure connection between your Connect:Direct bridge agent and the Connect:Direct node, you can use the SSL protocol or the TLS protocol.

Procedure

Configure a secure connection. For an example of how to do this, see [“Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node”](#) on page 53.

Mapping credentials for Connect:Direct

Map user credentials in Managed File Transfer to user credentials on a Connect:Direct node by using the default credential mapping function of the Connect:Direct bridge agent or by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping.

Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file

Map user credentials in Managed File Transfer to user credentials on Connect:Direct nodes by using the default credential mapping function of the Connect:Direct bridge agent. Managed File Transfer provides an XML file that you can edit to include your credential information.

About this task

After a Connect:Direct bridge agent has been created using the **fteCreateCDAgent** command, a `ConnectDirectCredentials.xml` file needs to be manually created. Before you can use a Connect:Direct bridge agent, you must edit this file to include host, user, and credential information. For more information, see [“Connect:Direct credentials file format”](#) on page 560. By default, this file is loaded from the current user's home directory, `/home/fteuser/ConnectDirectCredentials.xml` for example. To use another location, specify it using the `<credentialsFile>` element in the `ConnectDirectNodeProperties.xml` file.

Procedure

1. Ensure that the name attribute in the element `<tns:pnode name="Connect:Direct node host" pattern="wildcard">` contains the value of the name of the Connect:Direct node that the Connect:Direct bridge agent connects to. This value must be the same value that you specify for the **fteCreateCDAgent -cdNode** parameter.

The value of the `pattern` attribute can be either `wildcard` or `regex`. If this attribute is not specified, the default is `wildcard`.

2. Insert user ID and credential information into the file as child elements of `<tns:pnode>`.

You can insert one or more instances of the following `<tns:user>` element into the file:

```
<tns:user name="name"
          pattern="pattern"
          ignorecase="ignorecase"
          cdUserId="cdUserId"
          cdPassword="cdPassword"
          pnodeUserId="pnodeUserId"
          pnodePassword="pnodePassword">
</tns:user>
```

where:

- `name` is a pattern to match the MQMD user ID associated with the MFT transfer request.

- *pattern* specifies whether the pattern specified for the name attribute is a wildcard expression or a Java regular expression. The value of the *pattern* attribute can be either *wildcard* or *regex*. If this attribute is not specified, the default is *wildcard*.
- *ignorecase* specifies whether to treat the pattern specified by the name attribute as case sensitive. If this attribute is not specified, the default is *true*.
- *cdUserId* is the user ID that is used by the Connect:Direct bridge agent to connect to the Connect:Direct node specified by the name attribute of <tns:pnode> element. If possible, ensure that *cdUserId* is a Connect:Direct administrator user ID. If *cdUserId* cannot be a Connect:Direct administrator, ensure that the user ID has the following functional authorities at the Connect:Direct bridge node:
 - For a Windows node set the following authorities. This example is formatted with carriage returns to aid readability:

```
View Processes in the TCQ      value: yes
Issue the copy receive, copy send, run job, and run task Process statements
Issue the submit Process statement      value: yes
Monitor, submit, change, and delete all Processes      value: all
Access Process statistics      value: all
Use the trace tool or issue traceon and traceoff commands      value: yes
Override Process options such as file attributes and remote node ID      value: yes
```

- For a UNIX node set the following parameters in the *userfile.cfg* file:

```
pstmt.copy      value: y
pstmt.upload    value: y
pstmt.download  value: y
pstmt.runjob    value: y
pstmt.runtask   value: y
cmd.submit      value: y
pstmt.submit    value: y
cmd.chgproc     value: y
cmd.delproc     value: y
cmd.flsproc     value: y
cmd.selproc     value: a
cmd.selstats    value: a
cmd.trace       value: y
snode.ovrd     value: y
```

- *cdPassword* is the password associated with the user ID specified by the *cdUserId* attribute.
- You can optionally specify the *pnodeUserId* attribute. The value of this attribute is the user ID that is used by the Connect:Direct node specified by the name attribute of <tns:pnode> element to submit the Connect:Direct process. If you do not specify the *pnodeUserId* attribute, the Connect:Direct node uses the user ID specified by the *cdUserId* attribute to submit the Connect:Direct process.
- You can optionally specify the attribute *pnodePassword*. The value of this attribute is the password associated with the user ID specified by the *pnodeUserId* attribute.

If no user element matches the MQMD user ID, the transfer fails.

3. Optional: You can include one or more <tns:snode> elements as child elements of the <tns:user> element. The <tns:snode> element specifies credentials that are used by the Connect:Direct node that is part of the Connect:Direct bridge. These credentials are the user ID and password that

the Connect:Direct bridge node uses to connect to the Connect:Direct node that is the source or destination of the file transfer.

Insert one or many of the following elements into the file:

```
<tns:snode name="name"
           pattern="pattern"
           userId="userId"
           password="password" />
```

where:

- *name* is a pattern to match the name of the Connect:Direct node that is the source or destination of the file transfer.
- *pattern* specifies whether the pattern specified for the name attribute is a wildcard expression or a Java regular expression. The value of the pattern attribute can be either `wildcard` or `regex`. If this attribute is not specified, the default is `wildcard`.
- *userId* is the user ID that is used by the Connect:Direct node specified by the name attribute of the `<tns:pnode>` element to connect to a Connect:Direct node that matches the pattern specified by the name attribute of `<tns:snode>`.
- *password* is the password associated with the user ID specified by the `userId` attribute.

If no `<tns:snode>` element matches the secondary node of the file transfer, this does not cause the transfer to fail. The transfer is started and no user ID and password are specified for use with the `snode`.

Results

When searching for a pattern match for user names or Connect:Direct node names the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks

[“Configuring the Connect:Direct bridge” on page 135](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[“Connect:Direct credentials file format” on page 560](#)

The `ConnectDirectCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 384](#)

The `fteCreateCDAgent` command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

Mapping credentials for Connect:Direct by using exit classes

If you do not want to use the default credential mapping function of the Connect:Direct bridge agent, you can map user credentials in Managed File Transfer to user credentials on a Connect:Direct node by writing your own user exit. Configuring your own credential mapping user exits disables the default credential mapping function.

About this task

User exits that you create for mapping Connect:Direct credentials must implement the interface `com.ibm.wmqfte.exitroutine.api.ConnectDirectCredentialExit`. For more information, see [“CDCredentialExit.java interface” on page 855](#).

Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node

Configure the Connect:Direct bridge agent and the Connect:Direct node to connect to each other through the SSL protocol by creating a keystore and a truststore, and by setting properties in the Connect:Direct bridge agent properties file.

About this task

These steps include instructions for getting your keys signed by a certificate authority. If you do not use a certificate authority, you can generate a self-signed certificate. For more information about generating a self-signed certificate, see [Working with SSL or TLS on UNIX and Windows systems](#).

These steps include instructions for creating a new keystore and truststore for the Connect:Direct bridge agent. If the Connect:Direct bridge agent already has a keystore and truststore that it uses to connect securely to IBM MQ queue managers, you can use the existing keystore and truststore when connecting securely to the Connect:Direct node. For more information, see [“Configuring SSL or TLS encryption for MFT” on page 51](#).

Procedure

For the Connect:Direct node, complete the following steps:

1. Generate a key and signed certificate for the Connect:Direct node.

You can do this by using the IBM Key Management tool that is provided with IBM MQ. For more information, see [Working with SSL or TLS](#).

2. Send a request to a certificate authority to have the key signed. You receive a certificate in return.
3. Create a text file; for example, `/test/ssl/certs/CAcert`, that contains the public key of your certification authority.
4. Install the Secure+ Option on the Connect:Direct node.

If the node already exists, you can install the Secure+ Option by running the installer again, specifying the location of the existing installation, and choosing to install only the Secure+ Option.

5. Create a new text file; for example, `/test/ssl/cd/keyCertFile/node_name.txt`.
6. Copy the certificate that you received from your certification authority and the private key, located in `/test/ssl/cd/privateKeys/node_name.key`, into the text file.

The contents of `/test/ssl/cd/keyCertFile/node_name.txt` must be in the following format:

```
-----BEGIN CERTIFICATE-----
MIIcCnzCCAgigAwIBAgIBGjANBgkqhkiG9w0BAQUFADBEMQswCQYDVQQGEwJHqjES
MBAGA1UECBMJSjGFtcHNoaXJlMRAwDgYDVQQHEwdIdXJzbGV5M0wwCgYDVQQKEwNj
Qk0xDjAMBGNVBAstBU1RSVBUMQswCQYDVQDEwJDQTAeFw0xMTAzMDEwNjIwNDZa
Fw0yMTAyMjYxNjIwNDZaMFAxChZAJBgNVBAYTAkdCMRlWIAEYDVQQIEwLIYw1wc2hp
cmUxDDAKBGNVBAoTA01CTTEOMAwwGA1UECxMFTVFGVEUxZDZANBgNVBAMTBmJpbmJh
ZzCBnzANBgkqhkiG9w0BAQEFAA0BjQAwgYkCgYEAvgP1QIkLU9ypSKD1Xo0Do1yk
EymFXBOUpZrDvXjoSEC0vtWncJ199e+Vc4UpNybdyBu+Nkd1MnofX4QxeQcLAFj
WnhakqCiQ+JIAD5AurhnwChe0MV3kjA84GKH/±0SVqt1984mu/1DyS819XcfSSn
c00Msk1KbneVSCIV2XECaWAAAN7MHkwCQYDVR0TBAlwADA5Bg1ghkgBhvhCAQ0E
HxYdT3B1b1NTTCBHZW51cmF0ZWQgQ2VydG1maWNhdGUwHQYDVR00BBYEFNXMIpSc
csBXUniw4A3UzrZnCRsv3MB8GA1UdIwQYMBaAFDXY8rmj41Vz5+FVAoQb++cns+B4
MA0GCSqGSIb3DQEBBQUAA4GBAFc7k1Xa4pGKYgwchxKpE3ZF6FNWY4vBXS216/ja
8h/v18+iv010CL8t0ZOKSU95fyZLz0PKnCH7v+ItFSE3CIiEk9D1z2U6W091ICwn
17PL72TdfaL3kabwHYVf17IVcuL+VZsZ3HjLggP2qH09ZuJpSpet9+AxFVMLiaAb
8eHw
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,64A02DA15B6B6EF9

57kqxLOJ/gRU0IQ6hVK2YN13B4E1jAi1gSme0I5ZpEIG8CHXISKB7/0cke2FTqsV
1vI99QyCxSdWoMnt5fj51v7aPmVeS60b0m+U1Gre8B/Ze18JvJ204K2Uh72rDCXE
5e6eFxsDUM207sQDy20euBVELJtM2k0kL1R0doQsU3XQNgJw/t3ZIx5hPXWEQT
rjRQ064BEhb+PzzxPF8uwzZ9IruK9BJ/UUnqC60dBR87IEA4pnJD1Jvb2ML7EN9Z
5Y+50hTKI80GvBvWX04fHyvIX5as1whBoArXIS1AtNTprtPvoaP1zyIAeZ60Cvo/
Sfo+A2UhmteJe0JaZg2XZ3H495fAw/EHmjehzIACwukQ9nSIETgu4A1+CV64RJED
```

```
aYBCM8UjaAkbZDH5gn7+eBov0ssXAXWdyJBVhU0jXjvAj/e1h+kcSF1hax5D//AI
66nRMZzboSxNqkjcVd8wfdwP+bejDzUaaarJTS7lIFeLlw7eJ8MNAkMGicDkycL0
EPBU9X5QnHKLK0fYHN/1WgUk8qt3UytFXXfzTXGF3EbsWbBupkT5e5+1YcX80VZ6
SHFPNlHluCNy/riUcBy9iviVeodX8Iom0chSy05DKl8bwZNjYtUP+CtYHNFU5BaD
I+1uU0AeJ+wjQYKT1WaeIGZ3VxuNITJu18y5qDTXXfX7vxM50oWxa6U5+AYuGUMg
/itPZmUmNzHjTk7ghT6i1IQ0aBowXXKJB1Mmq/6BQXN2Ihkd9ys2qzvM1hdi5nAf
egmdiG50loLnBRqWbFR+DykpAhK4SaDi2F52Uxovw3Lhiw8dQP71zQ==
-----END RSA PRIVATE KEY-----
```

7. Start the Secure+ Admin Tool.

- On Linux or UNIX systems, run the command **spadmin.sh**.
- On Windows systems, click **Start > Programs > Sterling Commerce Connect:Direct > CD Secure+ Admin Tool**

The CD Secure+ Admin Tool starts.

8. In the CD Secure+ Admin Tool, double-click the **.Local** line to edit the main SSL or TLS settings.

- a) Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.
- b) Select **Disable Override**.
- c) Select at least one Cipher Suite.
- d) If you want two-way authentication, change the value of **Enable Client Authentication** to Yes.
- e) In the **Trusted Root Certificate** field, enter the path to the public certificate file of your certification authority, `/test/ssl/certs/CAcert`.
- f) In the **Key Certificate File** field, enter the path to the file that you created, `/test/ssl/cd/keyCertFile/node_name.txt`.

9. Double-click the **.Client** line to edit the main SSL or TLS settings.

- a) Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.
- b) Select **Disable Override**.

For the Connect:Direct bridge agent, perform the following steps:

10. Create a truststore. You can do this by creating a dummy key and then deleting the dummy key.

You can use the following commands:

```
keytool -genkey -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
```

```
keytool -delete -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
```

11. Import the public certificate of the certification authority into the truststore.

You can use the following command:

```
keytool -import -trustcacerts -alias myCA
-file /test/ssl/certs/CAcert
-keystore /test/ssl/fte/stores/truststore.jks
```

12. Edit the Connect:Direct bridge agent properties file.

Include the following lines anywhere in the file:

```
cdNodeProtocol=protocol
cdNodeTruststore=/test/ssl/fte/stores/truststore.jks
cdNodeTruststorePassword=password
```

In the example in this step, *protocol* is the protocol you are using, either SSL or TLS, and *password* is the password that you specified when you created the truststore.

13. If you want two-way authentication, create a key and certificate for the Connect:Direct bridge agent.

- a) Create a keystore and key.

You can use the following command:

```
keytool -genkey -keyalg RSA -alias agent_name
        -keystore /test/ssl/fte/stores/keystore.jks
        -storepass password -validity 365
```

- b) Generate a signing request.

You can use the following command:

```
keytool -certreq -v -alias agent_name
        -keystore /test/ssl/fte/stores/keystore.jks -storepass password
        -file /test/ssl/fte/requests/agent_name.request
```

- c) Import the certificate you receive from the preceding step into the keystore. The certificate must be in x.509 format.

You can use the following command:

```
keytool -import -keystore /test/ssl/fte/stores/keystore.jks
        -storepass password -file certificate_file_path
```

- d) Edit the Connect:Direct bridge agent properties file.

Include the following lines anywhere in the file:

```
cdNodeKeystore=/test/ssl/fte/stores/keystore.jks
cdNodeKeystorePassword=password
```

In the example in this step, *password* is the password that you specified when you created the keystore.

Related tasks

[“Configuring the Connect:Direct bridge” on page 135](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file

Specify which Connect:Direct process to start as part of a Managed File Transfer transfer. Managed File Transfer provides an XML file that you can edit to specify process definitions.

About this task

The **fteCreateCDAgent** command creates the file `ConnectDirectProcessDefinitions.xml` in the agent configuration directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name`. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

For each process that you want to specify to call as part of a transfer through the Connect:Direct bridge, perform the following steps:

Procedure

1. Define the Connect:Direct process that you want the Connect:Direct bridge agent to call as part of the transfer and save the process template in a file.
2. Open the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name/ConnectDirectProcessDefinitions.xml` file in a text editor.
3. Create a `<processSet>` element.

4. Inside the <processSet> element, create a <condition> element.
5. Inside the <condition> element, create one or more elements that define a condition that the transfer request must match to call the Connect:Direct process you defined in Step 1. These elements can be either <match> elements or <defined> elements.
 - Use a <match> element to specify that the value of a variable must match a pattern. Create the <match> element with the following attributes:
 - `variable` - the name of the variable whose value is compared. The variable is an intrinsic symbol. For more information, see [“Substitution variables for use with user-defined Connect:Direct processes”](#) on page 633.
 - `value` - the pattern to compare to the value of the specified variable.
 - Optional: `pattern` - the type of pattern used by the value of the `value` attribute. This pattern type can be `wildcard` or `regex`. This attribute is optional and the default is `wildcard`.
 - Use a <defined> element to specify that a variable must have a value defined. Create the <defined> element with the following attribute:
 - `variable` - the name of the variable that must have a value defined. The variable is an intrinsic symbol. For more information, see [“Substitution variables for use with user-defined Connect:Direct processes”](#) on page 633.

The conditions specified within the <condition> element are combined with a logical AND. All conditions must be met for the Connect:Direct bridge agent to call the process specified by this <processSet> element. If you do not specify a <condition> element, the process set matches all transfers.

6. Inside the <processSet> element, create a <process> element.
7. Inside the <process> element, create a <transfer> element.

The transfer element specifies the Connect:Direct process that the Connect:Direct bridge agent calls as part of the transfer. Create the <transfer> element with the following attribute:

- `process` - the location of the Connect:Direct process that you defined in step 1. The location of this file is specified with an absolute path or relative to the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name` directory.

Results

When searching for a condition match, the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks

[“Configuring the Connect:Direct bridge”](#) on page 135

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[“Connect:Direct process definitions file format”](#) on page 565

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)”](#) on page 384

The `fteCreateCDAgent` command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

Configuring MFT agents with MSCS

Managed File Transfer (MFT) agent Microsoft Cluster Service (MSCS) setup is supported, if the platform is one supported by MFT and running one of the versions of Windows.

About this task

This task describes two scenarios that you can follow to achieve failover of an MFT agent:

- Scenario 1: Configuring the agent as an MSCS resource.
- Scenario 2: Configuring the agent queue manager and the agent as MSCS resources.

Procedure

Scenario 1: Configuring the agent as an MSCS resource

- To configure the agent as an MSCS resource, complete the following steps:
 - a) Install Managed File Transfer locally on each machine in the cluster.
See [Installing Managed File Transfer](#).
 - b) Create the agent on the primary machine in the cluster.
The agent should be configured to connect to the agent queue manager using the CLIENT transport. Ensure that you create all of the objects on the queue manager for this agent. For information on how to do this, see [Setting up the agent](#).
 - c) Modify the agent to run as a Windows service, and configure it so that it does not start automatically when Windows restarts by setting the **Startup Type** field for the agent service in the Windows Services tool to Manual.
For more information, see [Starting an MFT agent as a Windows service](#).
 - d) Repeat step “2” on page 145 and step “3” on page 145 of Scenario 1 on the secondary machine.
This ensures that the file structure for logs, properties, and so on, exists on the other machine in the cluster. Note that there is no need to create the queue manager objects as in step “2” on page 145.
 - e) On the primary machine, add the agent as a 'Generic Service' under MSCS control.
To do this:
 - a. Right-click the cluster and select the **Role -> Add Resource -> 'Generic Service'**.
 - b. From the list of Windows services, select the agent service and complete the configuration wizard by clicking **Next**.
The agent service is now added as an MSCS resource. If a failover occurs, then the agent service will be started on the other machine.

Scenario 2: Configuring the agent queue manager and the agent as MSCS resources

- To configure the agent queue manager and the agent as MSCS resources, complete the following steps:
 - a) Configure the agent queue manager to run as an MSCS resource.
For information on how to do this, see [Putting a queue manager under MSCS control](#).
 - b) Create the agent on the primary machine in the cluster.
The agent should be configured to connect to the agent queue manager using the BINDINGS transport. Ensure that you create all of the objects on the queue manager for this agent. For information on how to do this, see [Setting up the agent](#).
 - c) Modify the agent to run as a Windows service, and configure it so that it does not start automatically when Windows restarts by setting the **Startup Type** field for the agent service in the Windows Services tool to Manual.

For more information, see [Starting an MFT agent as a Windows service](#).

- d) Ensure that the agent queue manager (which is under MSCS control) is running on the secondary machine.
The agent that is created on this machine will connect to the queue manager using the BINDINGS transport, and so needs to be available when the agent is created.
- e) Repeat step “2” on page 145 and step “3” on page 145 of Scenario 2 on the secondary machine.
This ensures that the file structure for logs, properties, and so on, exists on the other machine in the cluster. Note that there is no need to create the queue manager objects as in step “2” on page 145.
- f) Add the agent as a 'Generic Service' under MSCS control.
To do this:
 - a. Right-click the cluster and select the **Role -> Add Resource -> 'Generic Service'**.
 - b. From the list of Windows services, select the agent service and complete the configuration wizard by clicking **Next**.
- g) Modify the resource properties of the agent service to add the queue manager resource in the dependencies list.
This ensures that the queue manager resource is started before the agent is started.
- h) Take the queue manager resource offline and then bring the agent resource online. Verify if both the queue manager resource and the agent are started.
If a failover occurs, then the agent service and the agent queue manager will be started on the secondary machine.

Administering MFT

Use Managed File Transfer commands to administer Managed File Transfer. You can also use the IBM MQ Explorer for some of the administrative tasks.

Start transfer by placing a message in an agent command queue

You can also start a file transfer by putting a file transfer message on the command queue of the source agent. An example command queue name is `SYSTEM.FTE.COMMAND.AGENT01`. You must ensure that the message reaches the command queue of the correct source agent; if the message is received by an agent that does not match the source information in the XML, the message is rejected.

The transfer request XML must conform to the `FileTransfer.xsd` schema and use the `<request>` element as the root element. See [File transfer request message format](#) for information about the structure and content of a transfer request message. How you put the transfer request message on an agent command queue is task-specific. For example, you can use the IBM MQ Java API to put a message on the queue programmatically.

Starting an MFT agent

Before you can use a Managed File Transfer agent for a file transfer, you must first start the agent.

About this task

You can start a Managed File Transfer Agent from the command line. In this case, the agent process stops when you log off the system.

 On UNIX, Linux, and Windows, you can configure an agent so that it continues running when you log off from the system and can continue to receive file transfers.

 On z/OS, you can configure the agent to start as a started task from JCL without the need for an interactive session.

Note that, if an agent encounters an unrecoverable error when it is running, a first failure data capture (FDC) is generated and the agent is stopped.

Procedure

- To start an agent from the command line, use the **fteStartAgent** command.
For more information, see [“fteStartAgent” on page 505](#).
-  To configure an agent so that it continues running when you log off from the system:
 -  On Windows, configure the agent to run as a Windows service. For more information, see [“Starting an MFT agent as a Windows service” on page 147](#).
 -   On UNIX and Linux, configure the agent to start automatically during a reboot by using a script file. For more information, see [“Starting an MFT agent at UNIX and Linux system startup” on page 149](#).
-  On z/OS, configure the agent to start as a started task from JCL without the need for an interactive session.
For more information, see [“Starting an MFT agent on z/OS” on page 150](#).

Starting an MFT agent as a Windows service

You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

About this task

On Windows, when you start an agent from the command line, the agent process runs using the user name you used to log on to Windows. When you log off the system, the agent process stops. To prevent the agent stopping, you can configure an agent to run as a Windows service. Running as a Windows service also allows you to configure agents to be started automatically when the Windows environment starts or is restarted.

Complete the following steps to start an agent that runs as a Windows service. You must be running Managed File Transfer on one of the supported Windows versions to run the agent as a Windows service. For the list of supported environments, refer to the [IBM MQ System Requirements](#).

The exact steps depend on whether you have already created an agent or whether you are creating an agent. Both options are described in the following steps.

Procedure

1. If you are creating a Managed File Transfer agent, use the **fteCreateAgent**, **fteCreateCDAgent**, or **fteCreateBridgeAgent** command. Specify the **-s** parameter to run the agent as a Windows service. In the following example, the agent AGENT1 is created, which has an agent queue manager QMGR1. The Windows service runs using a user name of fteuser, which has an associated password ftepassword.

```
fteCreateAgent -agentName AGENT1 -agentQMGR QMGR1 -s -su fteuser -sp ftepassword
```

You can optionally specify a name for the service after the **-s** parameter. If you do not specify a name, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name you specified and *QMGR* is your agent queue manager name. In this example, the default name for the service is `mqmftAgentAGENT1QMGR1`.

Note: The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** rights. For information about how to configure this, see [“Guidance for running an MFT agent or logger as a Windows service”](#) on page 310.

For more information, see [“fteCreateAgent \(create an MFT agent\)”](#) on page 371, [“fteCreateCDAgent \(create a Connect:Direct bridge agent\)”](#) on page 384, or [“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)”](#) on page 375.

2. If you followed the previous step to create an agent, run the MQSC commands that are generated by the **fteCreateAgent**, **fteCreateCDAgent**, or **fteCreateBridgeAgent** command. These commands create the IBM MQ queues that are needed by the agent.

For example, for an agent named *AGENT1*, an agent queue manager named *QMGR1* and a coordination queue manager named *COORDQMGR1*, run the following command:

```
runmqsc QMGR1 MQ_DATA_PATH\mqft\config\COORDQMGR1\agents\AGENT1\AGENT1_create.mqsc
```

3. If you did not follow the previous steps to create an agent and instead want to configure an existing agent to run as a Windows service, first stop your agent if it is running, and then modify its configuration.

a) The following example uses an agent named *AGENT1*. Run the following command:

```
fteStopAgent AGENT1
```

b) Use the **fteModifyAgent** command to configure the agent to run as a Windows service:

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

For more information, see [“fteModifyAgent \(run an MFT agent as a Windows service\)”](#) on page 469.

4. Start your agent using the **fteStartAgent** command. Alternatively, you can use the Windows Services tool, which is available from Administrative Tools in the Control Panel, selected from the Windows desktop start menu, to start the service.

```
fteStartAgent AGENT1
```

The service continues to run even if you log off Windows. To ensure that the service also restarts when Windows restarts after a shutdown, the **Startup Type** field in the Windows Services tool is set to **Automatic** by default. Change this to **Manual** if you do not want the service to restart when Windows restarts.

5. Optional: To stop the agent, either use the **fteStopAgent** command or use the Windows Services tool. For example, from the command line, run the following command:

```
fteStopAgent AGENT1
```

- When you run the **fteStopAgent** command as a service, the command always runs using the **-i** parameter regardless of whether you specified this parameter. The **-i** parameter stops the agent immediately without completing any transfers that are in progress. This is caused by a limitation of the Windows service.

What to do next

If you have problems starting your Windows service, see [“Guidance for running an MFT agent or logger as a Windows service”](#) on page 310. This topic also describes the location of the Windows service log files.

Starting an MFT agent at UNIX and Linux system startup

A Managed File Transfer Agent can be configured to start at system startup on UNIX. When you log off a UNIX system your agent continues running and can receive file transfers.

When you have created and configured an agent using one of these Managed File Transfer commands; **fteCreateAgent**, **fteCreateCDAgent**, or **fteCreateBridgeAgent**, you can configure it to start automatically during a reboot on UNIX machines by using a script file that simply executes:

```
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

Where *mq_install_root* is the root directory of the required Managed File Transfer installation, the default is: `/opt/mqm` and *agent_name* is the name of the Managed File Transfer Agent to be started. The usage of this script file varies depending on the specific UNIX operating system.

Solaris

Solaris

On Solaris, follow these steps:

1. Run the following command, and keep track of the path returned:

```
which sh
```

For example, the path might be `/usr/bin/sh`

2. As the system root user ID, create your own `/etc/init.d/startmqmft` file.
3. Edit this file and add the script lines to it, using the returned path from step 1 as the first line in the script:

```
#!/usr/bin/sh
su mqmft_user mq_install_root/bin/fteStartAgent agent_name
```

4. Make the file executable, for example:

```
chmod 755 /etc/init.d/startmqmft
```

5. Symlink the file to the `rc3.d` directory:

```
ln -s /etc/init.d/startmqmft /etc/rc3.d/S98startmqmft
```

The prefix `S` means default state Started for Solaris. `98` is a sequence number. The suffix is the filename from `init.d`

HP-UX

HP-UX

On HP-UX, follow these steps:

1. Create a file called `/sbin/init.d/mqmft` with contents:

```
#!/bin/sh
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

2. Create a file called `/etc/rc.config.d/mqmft` with contents:

```
MQMFT=1
```

3. Symlink the file to the `rc3.d` directory and start the agent:

```
ln -s /sbin/init.d/mqmft /sbin/rc3.d/S84mqmft
```

Linux

Linux

For Linux systems there are multiple ways that you can start applications during the system boot process. In general, consider following these steps:

1. Create a file called `/etc/rc.mqmft` with contents:

```
#!/bin/sh
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

Where `mqmft_user` is the user ID under which the agent process is to run. This user ID must be a member of the `mqm` group.

2. Make the file executable, for example:

```
chmod 755 /etc/rc.mqmft
```

3. Next add the following line to `/etc/inittab`:

```
mqmft:5:boot:/etc/rc.mqmft
```

Other ways to start an agent during boot on Linux include adding the script lines to the `/etc/rc.d/rc.local` file, or on Linux SuSe, adding the script lines to the `/etc/init.d/boot.local` file. You should select the method that works best for your environment. Here is some more information on other ways to start an agent during startup on specific Linux distributions that are supported:

SLES 10 and 11

For SUSE Linux Enterprise Server (SLES) 10 and 11 systems, follow these steps:

1. As the system root user ID, create your own `/etc/init.d/rc.rclocal` file.
2. Add the following lines to the `rc.rclocal` file:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rc.rclocal
# Required-Start: $network $syslog
# Required-Stop: $network $syslog
# Default-Stop: 0 1 2 6
# Description: MQMFT agent startup
### END INIT INFO
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

3. Run the following commands:

```
chmod 755 rc.rclocal
chkconfig --add rc.rclocal
```

z/OS Starting an MFT agent on z/OS

On z/OS, in addition to running the `fteStartAgent` command from a UNIX System Services session, you can start an agent as a started task from JCL without the need for an interactive session.

A started task is used because it runs under a specific user ID and is not affected by users logging off.

Note:   Started tasks are typically run under an administrative user that might not have log-on privileges and so it is not possible to log on to the z/OS system as the user that the agent is running under. The **fteStartAgent**, **fteStopAgent**, and **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, cannot be issued for that agent.

From IBM MQ 9.0.2 and IBM MQ 9.0.0 Fix Pack 1, the agent property **adminGroup** is available for use with Managed File Transfer agents on z/OS. You can define a security manager group, for example MFTADMIN and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the **adminGroup** property to be the name of this security manager group.

```
adminGroup=MFTADMIN
```

Members of this group can then issue the **fteStartAgent**, **fteStopAgent**, and **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, for the agent that is running as a started task.

For more information, see the **adminGroup** property in [“The MFT agent.properties file” on page 526](#).

As a Java application, an agent is a UNIX System Services application that you can run from JCL by using the BFGAGSTP member, from a generated Managed File Transfer command PDSE library data set for an agent. For more information about how to create an MFT command PDSE library data set, and customize it for the required agent, see [“Creating an MFT Agent or Logger command data set” on page 62](#).

Enable agent remote connections with IBM MQ Advanced for z/OS, Value Unit Edition only

An MFT agent on z/OS, running under the product identifier (PID) of IBM MQ Advanced for z/OS VUE, can connect to a queue manager on z/OS using a client connection.

When an agent starts up, it writes a BFGPR0137I message to its event log (output0.log) showing the PID that it is running under. An example of this message is:

```
BFGPR0137I: Product usage data recording started for product 'MQ z/OS MFT', product id '5655-MF9'.
```

For details of IBM MQ products, their associated PID values, and export classifications, see [IBM MQ product identifiers and export information](#).

An MFT agent on z/OS, running under any other PID, can connect to a local queue manager using bindings connection only.

An MFT agent on z/OS can connect only to a queue manager, also running on z/OS, regardless of the MFT PID.

If an IBM MQ Advanced for z/OS VUE agent attempts to connect to a queue manager that is not running on z/OS, the following [BFGMQ1044E](#) message is issued, and agent start-up is ended:

```
BFGMQ1044E: Agent client connection on z/OS must be to a queue manager on z/OS
```

Related reference

[“Stopping an MFT agent on z/OS” on page 203](#)

If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Starting a new file transfer

You can start a new file transfer from the IBM MQ Explorer or from the command line and you can choose to transfer either a single file or multiple files in a group.

About this task

To start a new file transfer from the command line, see [fteCreateTransfer command](#).

To start a new file transfer by using the **Create New Managed File Transfer** wizard in IBM MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that the agent you want to use for the transfer is registered against. If you are currently connected to a coordination queue manager other than the one you want to use for the transfer, right-click that coordination queue manager name in the Navigator view and click **Disconnect**. Then right-click the name of the coordination queue manager you want to use and click **Connect**.
3. Start the **Create New Managed File Transfer** wizard by using either of the following methods:
 - a) Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to start the wizard.
 - b) Click **File > New > Other > Managed File Transfer Wizards > New Transfer Wizard**
4. Follow the instructions on the wizard panels. There is also context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press **Ctrl+F1** or **Shift+F1**.

Using transfer definition files

You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Transfer definition files are useful when you want to specify multiple source files and multiple destination files in a single transfer operation. You can use a transfer definition file to submit a complex file transfer. You can reuse and share the transfer definition file.

You can use two formats for a transfer definition file, and while these formats vary slightly, both conform to the `FileTransfer.xsd` schema. You can find this schema in the `samples\schema` directory of the Managed File Transfer installation.

The following two formats of transfer definition files are supported:

- A definition of the source and destination files for a transfer. This definition uses a `<transferSpecifications>` element as the root.
- A definition of the entire transfer, including source and destination files and the source and destination agents. This definition uses a `<request>` element as the root.
 - Files with this format can be generated from the **fteCreateTransfer** command by using the **-gt** parameter.

The following example shows a transfer definition file format that specifies only the source and destination files for a transfer:

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <item checksumMethod="MD5" mode="text">
    <source recursive="false" disposition="leave">
      <file>textTransferTest.txt</file>
    </source>
    <destination type="directory" exist="overwrite">
      <file>c:\targetfiles</file>
    </destination>
  </item>
</transferSpecifications>
```

To submit this format of transfer definition file you must specify the source and destination agents on the command line:

```
fteCreateTransfer -sa AGENT1 -sm agent1qm -da AGENT2 -dm agent2qm -td
c:\definitions\example1.xml
```

The following example is a transfer definition file format that specifies all information required for a transfer:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="3.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>fteuser</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="agent1qm"/>
    <destinationAgent agent="AGENT2" QMgr="agent2qm"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\*.jpg</file>
        </source>
        <destination type="directory" exist="error">
          <file>/targetfiles/images</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

You can generate a file with this format by using the **-gt** parameter on the **fteCreateTransfer** command. When you submit a transfer definition file with this format, you do not need to specify anything else on the command line:

```
fteCreateTransfer -td c:\definitions\example2.xml
```

You can override the source and destination agent information about the command line by passing in the normal parameters in addition to the transfer definition file. For example:

```
fteCreateTransfer -da AGENT9 -dm agent9qm -td c:\definitions\example2.xml
```

This example uses the command-line options to override the destination agent defined inside the transfer definition file with **AGENT9** and the destination queue manager defined in the transfer definition file as **agent9qm**.

Both of the formats described can contain one or more `<item>` elements. For further information about the `<item>` element, see [File transfer request message format](#). Each of these transfer items defines a source and destination file pair with additional attributes to control the behavior of the transfer. For example, you can specify the following behavior:

- Whether the transfer uses a checksum
- Whether the transfer is text or binary
- Whether to delete the source file after the transfer has completed
- Whether to overwrite the destination file if the file exists

An advantage of using transfer definition files is that you can specify additional options that are not available from the command line. For example, when you are carrying out message-to-file transfers, you can specify the `groupId` attribute by using a transfer definition file. This attribute specifies the IBM MQ group ID of the messages that are read from the queue. Another advantage of transfer definition files is that you can specify different options for each file pair. For example, you can specify whether a checksum is used, or whether the file is transferred in text or binary mode, on a file-by-file basis. If you use the command line, the same options apply for every file in a transfer.

For example:

```
<item checksumMethod="none" mode="binary">
  <source disposition="leave">
    <file>c:\sourcefiles\source1.doc</file>
  </source>
  <destination type="file" exist="error">
    <file>c:\destinationfiles\destination1.doc</file>
  </destination>
</item>

<item checksumMethod="MD5" mode="text">
  <source disposition="delete">
    <file>c:\sourcefiles\source2.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file encoding="UTF8" EOL="CRLF">c:\destinationfiles\destination2.txt</file>
  </destination>
</item>

<item checksumMethod="none" mode="text">
  <source recursive="false" disposition="leave">
    <file>c:\originfiles\source3.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file>c:\targetfiles\destination3.txt</file>
  </destination>
</item>
```

z/OS You can use items to transfer a file from a distributed system to a z/OS system:

```
<item checksumMethod="none" mode="text">
  <source recursive="false" disposition="leave">
    <file>textTransferTest.txt</file>
  </source>
  <destination type="dataset" exist="overwrite">
    <file encoding="IBM-1047">//TEXT.TRANS.TEST</file>
  </destination>
</item>
```

z/OS This example transfers the file `textTransferTest.txt` from the source agent to the data set `//TEXT.TRANS.TEST` on the destination agent in text mode. This transfer converts the source data from the default encoding of the source agent (no source encoding attribute is specified) to code page: IBM-1047.

Creating a scheduled file transfer

You can schedule a new file transfer either from the IBM MQ Explorer Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

About this task

You can set up a file transfer schedule to occur once, or to occur at the following intervals:

- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly

You can then specify the occurrences to stop at the following points:

- At a defined time and date
- After a defined number of occurrences

Alternatively, you can specify that the occurrences continue forever.

V9.0.0.9 If a scheduled transfer runs at the same time each day, use the **adjustScheduleTimeForDaylightSaving** attribute in the agent properties file to adjust the time the schedule takes place when the clocks change. See [The MFT agent .properties file](#) for more information.

To create a new scheduled file transfer using the command line, use the scheduling parameters (**-tb**, **-ss**, **-oi**, **-of**, **-oc**, and **-es**) for the `fteCreateTransfer` command.

To create a new scheduled file transfer using the **Create New Managed File Transfer** wizard in IBM MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that the agent you want to use for the transfer is registered against. If you are currently connected to a coordination queue manager other than the one you want to use for the transfer, right-click that coordination queue manager name in the Navigator view and click **Disconnect**. Then right-click the name of the coordination queue manager you want to use and click **Connect**.
3. Start the **Create New Managed File Transfer** wizard using either of the following methods:
 - a) Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to start the wizard.
 - b) Click **File > New > Other > Managed File Transfer Wizards > New Transfer Wizard**
4. Follow the instructions on the wizard panels. Ensure that you select the **Enable schedule transfer** check box and enter your schedule details on the **Schedule** tab. Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press Ctrl+F1 or Shift+F1.

Results

For information about the messages involved in scheduled file transfers, see [Message formats for scheduled transfers](#).

Working with pending file transfers

You can view scheduled file transfers that are pending from the IBM MQ Explorer. The **Pending Transfers** window displays all of the pending transfers registered with the coordination queue manager that you are currently connected to.

About this task

To view the status of a scheduled file transfer that has not yet started, use the following steps:

Procedure

1. Expand **Managed File Transfer** in the Navigator view. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Click **Pending Transfers**. The **Pending Transfers** window is displayed in the Content view.
4. The **Pending Transfers** window displays the following details about your scheduled file transfers:
 - a) **Name** The number of the scheduled file transfer. This number is automatically assigned.
 - b) **Source** The name of the source agent.
 - c) **Source File** The name of the file to be transferred on its host system.
 - d) **Destination** The name of the destination agent.
 - e) **Destination File** The name of the file after it is transferred to the destination system.
 - f) **Scheduled Start (selected time zone)** The time and date that the file transfer is scheduled to start in the administrator's selected time zone. To change the time zone displayed, click **Window > Preferences > IBM MQ Explorer > Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.
 - g) **Repeat Every** If you have chosen to repeat the scheduled transfer, the specified interval that you want to repeat the transfer, expressed as a number.
 - h) **Repeat Type** If you have chosen to repeat the scheduled transfer, the type of repeat interval you have specified for the file transfer. The type can be one of the following values: minutes, hours, days, weeks, months, or years.
 - i) **Repeat Until** If you have chosen to repeat the scheduled transfer, the details of when you want the repeating file transfer to stop. For example, a specified date and time, or after a specified number of occurrences.

Results

To refresh what is displayed in the **Pending Transfers** window, click the Refresh button  on the Content view toolbar.

To cancel a pending file transfer, right-click the particular transfer and click **Cancel**. Canceling a transfer completely discards the file transfer request.

Triggering a file transfer

You can set certain trigger conditions on a file transfer that must be true before that transfer can take place. If the triggering conditions are not true, the file transfer does not take place and a log message

is optionally submitted to record the fact the transfer did not happen. The file transfer request is then discarded. For example, you can set up a file transfer that takes place only if a named file on the system where the source agent is located is over a specified size, or if a particular named file exists on the system where the source agent is located. You can set up a triggered file transfer from either the IBM MQ Explorer or from the command line.

About this task

You can monitor a resource continually for a trigger condition to be satisfied. For further information about resource monitoring see: [“Monitoring MFT resources” on page 162](#).

There are three different triggering conditions that you can set. The conditions are as follows:

- If a particular file exists on the same system as the source agent
- If a particular file does not exist on the same system as the source agent
- If a particular file is over a certain size on the system where the source agent is located (the size can be expressed in bytes, KB, MB, or GB). These units of measurement use the 2¹⁰ convention, for example 1 KB equals 1024 bytes and 1 MB equals 1024 KB.

The triggering types in the preceding list can be combined in two ways:

- For a single condition, you can specify more than one file on the system where the source agent is located. This triggers the transfer if any one of the specified files meets the condition (Boolean operator OR).
- You can specify multiple conditions. This triggers the transfer only if all of the conditions are met (Boolean operator AND).

You can also combine a triggered transfer with a scheduled transfer. See [Creating a scheduled file transfer](#) for more information. In this case the trigger conditions are evaluated at the time the schedule is due to start, or for a repeating schedule every time the schedule is due to start.

Triggered transfers are not supported on protocol bridge agents.

To create a triggered file transfer by using the command line, use the **-tr** parameter on the [fteCreateTransfer](#) command.

To create a scheduled file transfer by using the **Create New Managed File Transfer** wizard in IBM MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Start the **Create New Managed File Transfer** wizard by using either of the following methods:
 - a) Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to open the wizard.
 - b) Click **File > New > Other > Managed File Transfer Wizards > New Transfer Wizard**
4. Follow the instructions on the wizard panels. Ensure that you select the **Enable triggered transfer** check box on the **Triggers** tab and complete the fields on that tab to set up triggering. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press **Ctrl+F1** or Shift+F1.

Monitoring file transfers that are in progress

You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in IBM MQ Explorer. This file transfer can be one started from either IBM MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

About this task

If you want to use IBM MQ Explorer to monitor transfers associated with a coordination queue manager on a remote system, follow the instructions in the [“Configuring IBM MQ Explorer to monitor a remote coordination queue manager” on page 159](#) topic.

Previous file transfer information is not retained after you stop and restart IBM MQ Explorer. At restart, the information about past transfers is cleared from the **Current Transfer Progress** tab. You can clear completed transfers using **Remove completed transfers**  at any point when IBM MQ Explorer is open.

Procedure

After you have started a new file transfer using IBM MQ Explorer or the command line, you can monitor the progress of your transfer in the **Current Transfer Progress** tab. The following information is displayed for each transfer in progress:

- a) **Source.** The name of the agent used to transfer the file from the source system.
- b) **Destination.** The name of the agent used to receive the file at the destination system.
- c) **Current file.** The name of the file currently being transferred. The part of the individual file that has already been transferred is displayed in B, KiB, MiB, GiB, or TiB along with total size of the file in parentheses. The unit of measurement displayed depends on the size of the file.
B is bytes per second. KiB/s is kibibytes per second, where 1 kibibyte equals 1024 bytes. MiB/s is mebibytes per second, where 1 mebibyte equals 1 048 576 bytes. GiB/s is gibibytes per second where 1 gibibyte equals 1 073 741 824 bytes. TiB/s is tebibytes per second where 1 tebibyte equals 1 099 511 627 776 bytes.
- d) **File number.** If you are transferring more than one file, this number represents how far through the total group of files the transfer is.
- e) **Progress.** The progress bar shows how complete the current file transfer is as a percentage.
- f) **Rate.** The rate the file is being transferred in KiB/s (kibibytes per second, where 1 kibibyte equals 1024 bytes.)
- g) **Started (selected time zone).** The time that the file transfer started, presented in the selected time zone of the administrator. To change the time zone displayed, click **Window > Preferences > IBM MQ Explorer > Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.

If the transfer enters a recovery state while transferring the file, the started time updates to reflect the time that the file transfer resumed.

Results

This tab regularly refreshes its information automatically, but to force a refreshed view of what is displayed in the **Current Transfer Progress** tab, click **Refresh**  on the Content view toolbar.

To delete file transfers from the **Current Transfer Progress** tab, click **Remove completed transfers**  on the Content view toolbar. Clicking this button removes file transfer details from the tab only; it does not stop or cancel a current or scheduled transfer.

If you want to return to the **Current Transfer Progress** tab after closing it, you can display the tab by clicking **Window > Show View > Other > Other > Managed File Transfer - Current Transfer Progress**. Click **OK**.

What to do next

Additionally, it is possible to develop applications for custom file transfer monitoring. This can be accomplished by creating a subscription to the appropriate Managed File Transfer administrative topic (either programmatically or administratively), and the monitor application can then receive Managed File Transfer file transfer activity publications on the topic. For more information on the subscription topic and publication message format, see [“File transfer progress message examples” on page 722](#).

Related tasks

[“Configuring IBM MQ Explorer to monitor a remote coordination queue manager” on page 159](#)

Use IBM MQ Explorer to monitor file transfers associated with a coordination queue manager running on a remote system. In IBM WebSphere MQ 7.5, or later, you require a system that is capable of running the IBM MQ Explorer. The IBM MQ Explorer component needs to be installed to be able to connect to the remote coordination queue manager.

[“Viewing the status of file transfers in the Transfer Log” on page 160](#)

You can view the details of file transfers by using the **Transfer Log** in IBM MQ Explorer. These can be transfers started from either the command line or the IBM MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Configuring IBM MQ Explorer to monitor a remote coordination queue manager

Use IBM MQ Explorer to monitor file transfers associated with a coordination queue manager running on a remote system. In IBM WebSphere MQ 7.5, or later, you require a system that is capable of running the IBM MQ Explorer. The IBM MQ Explorer component needs to be installed to be able to connect to the remote coordination queue manager.

About this task

Assumptions: Authority to connect to the remote coordination queue manager by configuring the queue manager to allow for remote connections.

For more information on how to configure this, see [“Connecting to a queue manager in client mode with channel authentication” on page 52](#) and [“Managing authorities for MFT-specific resources” on page 335](#).

To monitor queue managers and file transfers between agents on a system that is not running Windows or Linux, configure the IBM MQ Explorer to connect to the remote system using the following steps:

Procedure

1. Start the local IBM MQ Explorer.
2. When IBM MQ Explorer is loaded, right-click on the **Managed File Transfer** folder and select **New configuration**.
3. Proceed through the wizard, selecting the Coordination and Commands queue manager, then define a name for the configuration.
4. Click **finish** to complete the definition.
5. When the definition is finished, right-click on the definition and select **Connect**.

Results

Now start IBM MQ Explorer and use it to monitor transfer activity for the Managed File Transfer network associated with the coordination queue manager.

Related tasks

[“Monitoring file transfers that are in progress” on page 158](#)

You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in IBM MQ Explorer. This file transfer can be one started from either IBM MQ Explorer or

the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

[“Viewing the status of file transfers in the Transfer Log” on page 160](#)

You can view the details of file transfers by using the **Transfer Log** in IBM MQ Explorer. These can be transfers started from either the command line or the IBM MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Viewing the status of file transfers in the Transfer Log

You can view the details of file transfers by using the **Transfer Log** in IBM MQ Explorer. These can be transfers started from either the command line or the IBM MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Procedure

1. Expand **Managed File Transfer** in the Navigator view and then expand the name of the coordination queue manager that you want to view the transfer log for.
2. Click **Transfer Log** in the Navigator view. The **Transfer Log** is displayed in the Content view.
3. The **Transfer Log** window displays the following details about your file transfers:
 - a) **Source** The name of the agent on the system where the source file is located.
 - b) **Destination** The name of the agent on the system you want to transfer the file to.
 - c) **Completion State** The status of the file transfer. The state can be one of the following values: "Started", "In progress", "Successful", "Partially Successful", "Cancelled", or "Failed".
 - d) **Owner** The user ID on the host that submitted the transfer request.
 - e) **Started (selected time zone)** The time and date that the file transfer request was accepted by the Managed File Transfer agent, presented in the selected time zone of the administrator. To change the time zone displayed, click **Window > Preferences > IBM MQ Explorer > Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.
 - f) **State Recorded (selected time zone)** (This column is not displayed by default. You can choose to display the column by using the **Configure Transfer Log Columns**  window.) The time and date that the completion state was recorded, in the time zone selected by the administrator.
 - g) **Job Name** An identifier specified by the user by using the **-jn** parameter of **fteCreateTransfer** or in an Ant script
 - h) **Transfer ID** The unique identifier for the file transfer.
 - i) **Connect: Direct** Details about **Process Number**, **Process Name**, **Primary Node**, **Secondary Node**, **Source Type** and **Destination Type** are listed.

Results

Note: The internal format of the Transfer Log was changed in IBM MQ 8.0.0 Fix Pack 1 for APAR IC99545. As a result, if an IBM MQ Explorer is upgraded to 8.0.0.1 or later, and then restored to 8.0.0.0, no audit XML is displayed for transfers that took place while IBM MQ Explorer was at 8.0.0.1. The XML panel in the **Properties** window for these transfers will contain an empty text box.

To view further details about a completed transfer, expand the transfer that you are interested in by clicking the plus sign (+). You can then see all of the source and destination file names included in that transfer. However, if the transfer is currently in progress and consists of many files, you can view only the files that have already been transferred so far.

To refresh what is displayed in the **Transfer Log**, click the **Refresh** button  on the Content view toolbar. The file transfer information in the Transfer Log remains in the log after you stop and restart the IBM MQ Explorer. If you want to delete all completed file transfers from the log, click **Remove Completed Transfers**  on the Content view toolbar.

To delete an individual completed file transfer from the log, right-click the transfer and click **Delete**. If you delete a transfer, it does not stop or cancel a transfer that is in progress or that has been scheduled; you are deleting only the stored historical data.

To copy the unique identifier of a transfer to the clipboard, right-click that transfer and click **Copy ID**.

The metadata and the complete audit XML for the transfer are available from the pop-up menu, under the **Properties** action.

Related concepts

[“Timeout option for file transfers in recovery” on page 236](#)

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

Related tasks

[“Monitoring file transfers that are in progress” on page 158](#)

You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in IBM MQ Explorer. This file transfer can be one started from either IBM MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

[“Configuring the Transfer Log” on page 161](#)

You can configure what information is displayed and how information is displayed in the **Transfer Log** in the IBM MQ Explorer.

Configuring the Transfer Log

You can configure what information is displayed and how information is displayed in the **Transfer Log** in the IBM MQ Explorer.

About this task

To rearrange the order of the columns in the **Transfer Log**, click the title of the column you want to move and drag the column to its new position. The new column order is retained only until you next stop and restart the IBM MQ Explorer.

To filter entries in the **Transfer Log**, enter a string in the **Filter the displayed log entries** field. To restore all of the entries to the log, delete the string you entered from the field. You can use any valid Java regular expression in this field. For more information, see [“Regular expressions used by MFT” on page 633](#).

To customize which columns are displayed in the Transfer Log, use **Configure Transfer Log Columns** . Use the following steps to start and use the **Configure Transfer Log Columns** window.

Procedure

1. Ensure that you have the **Transfer Log** open in the Content view. Click **Configure Transfer Log Columns**  on the Content view toolbar. The **Configure Transfer Log Columns** window opens.
2. To customize your view of the **Transfer Log**, select or clear individual check boxes for the columns you want to show or hide. You can click **Select All**, then **OK** to select all of the check boxes or **Deselect All**, then **OK** to clear all of the check boxes.

Related tasks

[“Monitoring file transfers that are in progress” on page 158](#)

You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in IBM MQ Explorer. This file transfer can be one started from either IBM MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

[“Viewing the status of file transfers in the Transfer Log” on page 160](#)

You can view the details of file transfers by using the **Transfer Log** in IBM MQ Explorer. These can be transfers started from either the command line or the IBM MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Monitoring MFT resources

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

About this task

Managed File Transfer resource monitoring uses the following terminology:

monitor

A process that polls a resource (such as a directory or queue) at a predefined regular interval to see if the resource contents have changed. If they have, the contents are compared with the set of conditions for this monitor. If there is a match, the task for this monitor is started.

resource

The system resource the monitor examines every poll interval to be compared with the trigger conditions. Queues, directories, or nested directory structures can be the monitored resource.

condition

An expression that is evaluated (typically against the content of the monitored resource). If the expression evaluates to true, the condition contributes to the overall trigger condition.

trigger condition

The overall condition, which is satisfied when all conditions are satisfied. When the trigger condition is satisfied the task can proceed.

task

The operation that is started when the trigger condition or set of conditions is satisfied. Supported tasks are file transfer and command call.

trigger file

A file that is placed in a monitored directory to indicate that a task (typically a transfer) can begin. For example, it might indicate that all the files to be processed have arrived in a known location and can be transferred or otherwise acted upon. The name of the trigger file can be used to specify the files to be transferred by using variable substitution. For more information, see [“Customizing MFT resource monitor tasks with variable substitution” on page 172](#).

The trigger file is also known as ready file or go file. However, in this documentation it is always referred to as the trigger file.

Resource monitoring is not supported on protocol bridge agents or Connect:Direct bridge agents.

MFT resource monitoring concepts

An overview of the key concepts of the Managed File Transfer resource monitoring feature.

Monitors

You create a resource monitor by using the **fteCreateMonitor** command, which creates and starts a new resource monitor from the command line. The resource monitor is associated with a Managed File Transfer agent, and is only active when that agent is started. When the monitoring agent stops, so does the monitor. If the agent is already started when the monitor is created, the monitor starts immediately. The monitoring agent must also be the source agent of the task that is initiated by the monitor.

Monitor names must be unique within their agent. The monitor name must be a minimum of one character in length and must not contain asterisk (*), percent (%) or question mark (?) characters. The case of supplied monitor names is ignored and the monitor name is converted to uppercase. If you try to create

a monitor with a name that is already present, the request is ignored and the attempt is logged to the monitor log topic.

Note: **V9.0.4** You cannot create a resource monitor with a task definition that contains scheduled transfers.

There is no restriction on the number of monitors that can be created on an agent, and all run with the same priority. Consider the implications of overlapping monitored resources, conflicting trigger conditions and how frequently the resources are polled.

Overlapping resource monitors can cause:

- Possible contention on the source location/items.
- Possible duplicate transfer requests for same source items.
- Unexpected errors or failures for transfers due to conflicts of source items.

If multiple monitors scan the same location and can trigger on the same items, you can potentially end up with the problem of two different monitors submitting managed transfer requests for the same item.

Resource monitors look at the contents of resources after every poll interval period. The contents of the resource are compared with the trigger conditions and if those conditions are satisfied, the task associated with the monitor is called.

The task is started asynchronously. If there is a condition match, and the task is started, the monitor continues to poll for further changes to the resource contents. So for example, if a match occurred because a file called `reports.go` arrived in a monitored directory, the task would be started once. At the next poll interval, even if the file still exists, the task is not started again. However, if the file is deleted and then placed in the directory again, or the file is updated (such that the last modified date attribute is changed), the next trigger condition check causes the task to be called again.

Resources

Resource monitors in Managed File Transfer can poll the contents of the following two types of resource:

Directories or nested directory structures

A common scenario is to monitor a directory for the presence of a trigger file. An external application might be processing multiple files and placing them in a known source directory. When the application has completed its processing, it indicates that the files are ready to be transferred, or otherwise acted upon, by placing a trigger file into a monitored location. The trigger file can be detected by a Managed File Transfer resource monitor and the transfer of those files from the source directory to another Managed File Transfer Agent is initiated.

By default, the specified directory is monitored. To also examine sub-directories set the recursion level in the **fteCreateTransfer** command.

Two examples of monitoring a directory are as follows:

- Monitor for a trigger file (for example `trigger.file`) and then transfer a wildcard (for example, `*.zip`).
- Monitor for `*.zip` and then transfer `${FilePath}` (for example, the file that triggered the transfer). For more information about variable substitution, see [“Customizing MFT resource monitor tasks with variable substitution”](#) on page 172.

Note: Do not create a monitor that monitors for `*.zip`, and then transfers `*.zip`. The monitor tries to start a transfer of `*.zip` for every `.zip` file on your system. That is, the monitor generates * number of transfers for `*.zip`.

For an example of creating a resource monitor to monitor a directory, see [“Monitoring a directory and using variable substitution”](#) on page 170.

IBM MQ queues

An example of monitoring a queue is that an external application might be generating messages and placing them on a known queue with the same group ID. When the application has completed putting messages on the queue, it indicates that the group is complete. The complete group of messages can

be detected by a Managed File Transfer resource monitor and the transfer of the group of messages from the source queue to a file is initiated. For an example of creating a resource monitor to monitor a queue, see [“Example: Configuring an MFT resource” on page 171](#).

Note: You can specify only one monitor per queue. If you specify more than one monitor to poll an IBM MQ queue, unpredictable behavior occurs.

Monitoring data sets is not supported.

Trigger conditions

The condition is met when the resource contains a value that matches some other string or pattern. Conditions can be one of the following:

- Match on file name (pattern)
- No match on file name (pattern)
- File size
- Match if file size remains the same for a number of polls

File name matching can be expressed as:

- Exact string match
- Simple wildcard match as described in [“Using wildcard characters with MFT” on page 627](#)
- Regular expression match

File names can also be excluded from file name matching by using a wildcard or Java regular expression that identifies file names that are never matched.

When a matching file is detected, its last modified time stamp is retained. If subsequent polls detect that the file has been changed, the trigger condition is satisfied again, and the task is started. If the condition is to detect when a file does not exist, if no file in the monitored directory matches the file name pattern, the task is started. If a file is then added to the directory that does match the file name pattern, the task is only started if the file is then deleted.

Tasks

Managed File Transfer supports the following two types of task that you can configure to be started by resource monitors:

- File transfer
- Command

File transfer tasks are defined in the same way as any other file transfer. A useful way to generate the task XML required by a monitor is to run the [fteCreateTransfer](#) command with the **-gt** parameter. This command generates a task definition as an XML document, including the transfer specification. You then pass the name of the task XML document as the value for the **-mt** parameter on the [fteCreateMonitor](#) command. When the **fteCreateMonitor** is run, it reads the task XML document. After the **fteCreateMonitor** is run, any changes that are made to the task XML file are not used by the monitor.

Command tasks can run Ant scripts, call executable programs, or run JCL jobs. For more information, see [Configuring monitor tasks to invoke commands and scripts](#).

When using a file transfer task, you can select how many trigger conditions are batched into a task. The default is for one trigger condition to start one task. You can run the [fteCreateMonitor](#) command with the **-bs** option to select the number of trigger conditions that are batched together into one task.

Backing up and restoring resource monitors

You can back up the resource monitors that you have already defined so that you can re-use them in the future. There are various options that you can use as follows:

- Use the **fteCreateMonitor** command with the **-ox** parameter to export a resource monitor configuration to an XML file, and with the **-ix** parameter to restore a resource monitor by importing the resource monitor configuration from an XML file.
- Use the **fteListMonitors** command with the **-ox** to export the definition for a single resource monitor to an XML file.
- **V 9.0.5** From IBM MQ 9.0.5, use the **fteListMonitors** command with the **-od** to export multiple resource monitor definitions to a specified directory. Each resource monitor definition is saved to separate XML file. You can also use the **-od** option to export a single resource monitor definition to a specified directory.

For more information, see [“Backing up and restoring MFT resource monitors” on page 184.](#)

Resource monitor logging

V 9.0.3

From IBM MQ 9.0.3, Managed File Transfer includes resource monitor logging. For more information, see [“Logging MFT resource monitors” on page 182.](#)

Related concepts

[“Customizing MFT resource monitor tasks with variable substitution” on page 172](#)

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Related tasks

[“Configuring MFT monitor tasks to start commands and scripts” on page 166](#)

Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

[“Example: Configuring an MFT resource” on page 171](#)

You can specify an IBM MQ queue as the resource to be monitored by a resource monitor by using the **-mq** parameter with the **fteCreateMonitor** command.

[“Monitoring a queue and using variable substitution” on page 177](#)

You can monitor a queue and transfer messages from the monitored queue to a file by using the **fteCreateMonitor** command. The value of any IBM MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

Related reference

[“fteCreateMonitor: create an MFT resource monitor” on page 394](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

[“fteListMonitors: list MFT resource monitors” on page 455](#)

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

[“fteDeleteMonitor: delete an MFT resource monitor” on page 446](#)

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

Configuring MFT monitor tasks to start commands and scripts

Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

About this task

The file path to the executable program, Ant script, or JCL job that you want the monitoring agent to call must be included in the `commandPath` of the monitoring agent. For information about the command path property, see [“commandPath MFT property” on page 347](#).

You can create the task definition XML document in one of the following ways:

- Create the task definition XML document manually according to the `FileTransfer.xsd` schema. For more information, see [“Create the task definition XML manually according to the schema” on page 166](#).
- Edit the XML document generated by the **fteCreateTransfer -gt** parameter as the basis for your task definition. For more information, see [“Creating a task definition document by modifying a generated document” on page 168](#).

Whether you want a transfer task or a command task, the task definition must start with a `<request>` root element. The child element of `<request>` must be either `<managedTransfer>` or `<managedCall>`. You would typically choose `<managedCall>` when there is a single command or script to run, and `<managedTransfer>` if you want the task to include a file transfer and optionally up to four command calls.

Create the task definition XML manually according to the schema

About this task

You can manually create a task definition XML file according to the schema `FileTransfer.xsd`. This schema can be found in the `MQ_INSTALLATION_PATH/mqft/samples/schema`. For more information about this schema, see [“File transfer request message format” on page 755](#).

Example

The following example shows an example task definition XML document saved as `cleanuptask.xml`, which uses the `<managedCall>` element to call an Ant script called `RunCleanup.xml`. The `RunCleanup.xml` Ant script must be located on the `commandPath` of the monitoring agent.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>hostName</hostName>
      <userID>userID</userID>
      <mqmdUserID>mqmdUserID</mqmdUserID>
    </originator>
    <agent QMgr="QM1" agent="AGENT1"/>
    <reply QMGR="QM1">reply</reply>
    <transferSet priority="1">
      <metaDataSet>
        <metaData key="name1">value1</metaData>
      </metaDataSet>
      <call>
        <command name="RunCleanup.xml" type="antscript" retryCount="2"
          retryWait="30" successRC="0">
          <target>check_exists</target>
          <target>copy_to_archive</target>
          <target>rename_temps</target>
          <target>delete_files</target>
          <property name="trigger.filename" value="{FileName}"/>
        </call>
      </transferSet>
    </managedCall>
  </request>
```

```

        <property name="trigger.path" value="{FilePath}"/>
    </command>
</call>
</transferSet>
<job>
  <name>JOBCLEAN1</name>
</job>
</managedCall>
</request>

```

The `<agent>` element specifies the Managed File Transfer Agent that is configured with the named Ant script on its `commandPath`.

The `<call><command>` . . . structure defines the executable or script you want to run. The command takes an optional `type` attribute that can have one of the following values:

antscript

Run an Ant script in a separate JVM.

executable

Invoke an executable program.

jcl

Invoke a JCL job.

If you omit the `type` attribute, the default value `executable` is used.

The `name` attribute specifies the name of the Ant script, executable, or JCL job you want to run, without any path information. The agent searches for the script or program in the locations specified by the `commandPath` property in the agent's `agent.properties` file.

The `retrycount` attribute specifies the number of times to try calling the program again if the program does not return a success return code. The value assigned to this attribute must not be negative. If you do not specify the `retrycount` attribute, a default value of zero is used.

The `retrywait` attribute specifies the time to wait, in seconds, before trying the program invocation again. The value assigned to this attribute must not be negative. If you do not specify the `retrywait` attribute, a default value of zero is used.

The `successrc` attribute is an expression used to determine when the program invocation successfully runs. The process return code for the command is evaluated using this expression. The value can be composed of one or more expressions combined with a vertical bar (`|`) character to signify Boolean OR, or an ampersand (`&`) character to signify Boolean AND. Each expression can be one of the following types of expression:

- A number to indicate an equality test between the process return code and the number.
- A number prefixed with a greater than character (`>`) to indicate a greater-than test between the number and the process return code.
- A number prefixed with a less than character (`<`) to indicate a less-than test between the number and the process return code.
- A number prefixed with an exclamation point character (`!`) to indicate a not-equal-to test between the number and the process return code. For example: `>2&<7&!5|0|14` is interpreted as the following return codes being successful: 0, 3, 4, 6, 14. All other return codes are interpreted as being unsuccessful.

If you do not specify the `successrc` attribute, a default value of zero is used. This means that the command is judged to have successfully run if, and only if, it returns a code of zero.

For an Ant script, you would typically specify `<target>` and `<property>` elements. The `<target>` element values must match the target names in the Ant script.

For executable programs, you can specify `<argument>` elements. Nested argument elements specify arguments to pass to the program that is being called as part of the program invocation. The program arguments are built from the values specified by the argument elements in the order that the argument elements are encountered. You can specify zero or more argument elements as nested elements of a program invocation.

The administrator defines and starts the monitor as normal using the task definition XML document that includes the `<managedCall>` element. For example:

```
fteCreateMonitor -ma AGENT1 -mm QM1 -md /monitored -mn MONITOR01 -mt
/tasks/cleanuptask.xml -pi 30 -pu seconds -tr match,*go
```

The path to the transfer definition XML document must be on the local file system that you run the **fteCreateMonitor** command from (in this example `/tasks/cleanuptask.xml`). The `cleanuptask.xml` document is used to create the resource monitor only. Any tasks that the `cleanuptask.xml` document references (Ant scripts or JCL jobs) must be in the command path of the monitoring agent. When the monitor trigger condition is satisfied, any variables in the task definition XML are substituted with actual values from the monitor. So for example `${FilePath}` is replaced in the request message sent to the agent with `/monitored/cleanup.go`. The request message is put on the agent command queue. The command processor detects that the request is for a program call and starts the specified program. If a command of type `antscript` is called, a new JVM is started and the Ant task runs under the new JVM. For more information about using variable substitution, see [Customizing tasks with variable substitution](#).

Related concepts

[“Customizing MFT resource monitor tasks with variable substitution” on page 172](#)

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

[“commandPath MFT property” on page 347](#)

Use the `commandPath` property to specify the locations that Managed File Transfer can run commands from. Take extreme care when you set this property because any command in one of the specified `commandPaths` can effectively be called from a remote client system that is able to send commands to the agent.

Creating a task definition document by modifying a generated document

About this task

You can create the monitor task definition document by modifying the XML document generated by the `-gt` option of **fteCreateTransfer**. The generated document has a `<request>` followed by `<managedTransfer>` element. To convert this task definition to a valid `<managedCall>` structure, follow these steps:

Procedure

1. Replace the `<managedTransfer>` start and end tags with `<managedCall>` tags.
2. Remove any `<schedule>` element and child nodes.
3. Replace the `<sourceAgent>` start and end tags with `<agent>` to match the monitoring agent configuration details.
4. Remove `<destinationAgent>` and `<trigger>` elements.

5. Remove `<item>` elements.
6. Insert a new `<call>...</call>` structure within the `<transferSet>` element. This structure contains the command definition as shown in the following example:

```
<call>
  <command name="RunCleanup.xml" type="antscript" retryCount="2"
    retryWait="30" successRC="0">
    <target>check_exists</target>
    <target>copy_to_archive</target>
    <target>rename_temps</target>
    <target>delete_files</target>
    <property name="trigger.filename" value="{FileName}"/>
    <property name="trigger.path" value="{FilePath}"/>
  </command>
</call>
```

Example

You can also retain the `<managedTransfer>` element including all the file transfer details, and insert up to four command calls. In this case you insert any selection of the following call elements between the `<metaDataSet>` and `<item>` elements:

preSourceCall

Call a program on the source agent before starting the transfer.

postSourceCall

Call a program on the source agent after completing the transfer.

preDestinationCall

Call a program on the destination agent before starting the transfer.

postDestinationCall

Call a program on the destination agent after completing the transfer.

Each of these elements takes the `<command>` element structure as described in the earlier example. The `FileTransfer.xsd` schema defines the types used by the various call elements.

The following example shows `preSourceCall`, `postSourceCall`, `preDestinationCall`, and `postDestinationCall` in a task definition document:

```
:
<transferSet priority="1">
  <metaDataSet>
    <metaData key="key1">value1</metaData>
  </metaDataSet>
  <preSourceCall>
    <command name="send.exe" retryCount="0" retryWait="0" successRC="0"
      type="executable">
      <argument>report1.pdf</argument>
      <argument>>true</argument>
    </command>
  </preSourceCall>
  <postSourceCall>
    <command name="//DO_IT.JCL" retryCount="0" retryWait="0" successRC="0"
      type="jcl">
      <argument>argument</argument>
    </command>
  </postSourceCall>
  <preDestinationCall>
    <command name="ant_script.xml" retryCount="0" retryWait="0" successRC="0"
      type="antscript">
      <target>step1</target>
      <property name="name" value="value"/>
    </command>
  </preDestinationCall>
  <postDestinationCall>
    <command name="runit.cmd" retryCount="0" retryWait="0" successRC="0"/>
  </postDestinationCall>
  <item checksumMethod="none" mode="binary">
:
```

You can mix different types of command into the transfer. Argument, target, and property elements are optional.

Monitoring a directory and using variable substitution

You can monitor a directory using the **fteCreateMonitor** command. The value of a substitution variable can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called AGENT_HOP. The directory that AGENT_HOP monitors is called /test/monitored. The agent polls the directory every 5 minutes.

After a .zip file is written to the directory, the application that writes the file to the directory writes a trigger file to the same directory. The name of the trigger file is the same as the name of the .zip file, but has a different file extension. For example, after the file file1.zip is written to the directory, the file file1.go is written to the directory. The resource monitor monitors the directory for files that match the pattern *.go then uses variable substitution to request a transfer of the associated .zip file.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>blue.example.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP"/>
    <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP"/>
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <file>/test/monitored/_${fileName}{token=1}{separator=.}.zip</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/out/_${fileName}{token=1}{separator=.}.zip</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The variables that are replaced with the values associated with the trigger file are highlighted in **bold**. This task XML is saved to the file /home/USER1/task.xml

2. Create a resource monitor to monitor the directory /test/monitored.

Submit the following command:

```
fteCreateMonitor -ma AGENT_HOP -mm QM_HOP -md /test/monitored
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr match,*.go -pi 5 -pu minutes
```

3. A user or program writes the file jump.zip to the directory /test/monitored, then writes the file jump.go to the directory.
4. The monitor is triggered by the existence of the file jump.go. The agent substitutes the information about the trigger file into the task XML.

This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
```

```

        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedTransfer>
  <originator>
    <hostName>blue.example.com</hostName>
    <userID>USER1</userID>
  </originator>
  <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP"/>
  <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP"/>
  <transferSet>
    <item mode="binary" checksumMethod="none">
      <source>
        <file>/test/monitored/jump.zip</file>
      </source>
      <destination type="file" exist="overwrite">
        <file>/out/jump.zip</file>
      </destination>
    </item>
  </transferSet>
</managedTransfer>
</request>

```

Results

The transfer defined by the task XML is performed. The `jump.zip` file is read from the `/test/monitored` directory by `AGENT_HOP` and is transferred to a file called `/out/jump.zip` located on the system where `AGENT_SKIP` is running.

Related concepts

[“Customizing MFT resource monitor tasks with variable substitution” on page 172](#)

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Related tasks

[“Configuring MFT monitor tasks to start commands and scripts” on page 166](#)

Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

Related reference

[“fteCreateMonitor: create an MFT resource monitor” on page 394](#)

The **`fteCreateMonitor`** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

Example: Configuring an MFT resource

You can specify an IBM MQ queue as the resource to be monitored by a resource monitor by using the **`-mq`** parameter with the **`fteCreateMonitor`** command.

About this task

In this example, the resource to be monitored is the queue `MONITORED_QUEUE`. This queue must be on the monitoring agent's queue manager, `QM_NEPTUNE`. The condition that the queue is monitored for is the presence of a complete group of messages. The task to be performed if the condition is satisfied is defined in the file `task.xml`.

Note: Do not create more than one resource monitor to monitor an individual queue. If you do then unpredictable behavior occurs.

Procedure

Type the following command:

```
fteCreateMonitor -ma AGENT_NEPTUNE -mn myMonitor -mm QM_NEPTUNE -mq MONITORED_QUEUE  
-mt task.xml -tr completeGroups -pi 5 -pu minutes
```

The monitor checks the queue every five minutes to see if the condition `completeGroups` is true. If there are one or more complete groups on the queue, the monitor runs the task defined in the `task.xml` file once for each complete group.

Customizing MFT resource monitor tasks with variable substitution

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.



Attention: Variable names are not case-sensitive.

The variables that are used for substitution are only available for positive trigger conditions. Only `match` and `fileSize` trigger conditions cause variables to be substituted. If a `noMatch` condition is used, and there are substitution variable names in the task definition, the task is not called, and the monitor raises a return code of 110 and error message `BFGDM0060E`.

If the monitored resource is a queue

The value of any IBM MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition.

User-defined message properties are prefixed with `usr.` but do not include this prefix in the variable name. Variable names must be preceded by a dollar sign (\$) character and enclosed in braces {}.

For example, `${destFileName}` is replaced with the value of the `usr.destFileName` message property of the first message to be read from the source queue. For more information, see [“IBM MQ message properties read by MFT from messages on source queues” on page 653](#) and [“Monitoring a queue and using variable substitution” on page 177](#).

If a variable is not defined as a message property the monitor reports a `BFGDM0060E` error and returns return code 110 (Monitor task variable substitution failed). In addition to this, the agent writes the following error message to its event log (`outputN.log`):

```
BFGDM0113W: Trigger failure for <monitor name> for reason BFGDM0060E: A monitor task could  
not complete as a variable substitution <variable name> was not present.
```

If moderate or verbose resource monitor logging is enabled for the monitor, the monitor writes the following message to the resource monitor event log of the agent (`resmoneventN.log`):

```
BFGDM0060E: A monitor task could not complete as a variable substitution  
<variable name> was not present.
```

See [“Logging MFT resource monitors” on page 182](#) for more information on resource monitor logging.

The following table shows which substitution variables are provided by default. For example, `{AGENTNAME}` is replaced with the name of the resource monitor agent.

Variable	Description
AGENTNAME	The name of the resource monitor agent.

Table 18. Substitution variables provided by default (continued)

Variable	Description
QUEUENAME	The name of the queue that is being monitored.
ENCODING	The character encoding of the first message on the queue or the first message in a group.
MESSAGEID	The IBM MQ message ID of the first message on the queue or the first message in the group.
GROUPID	The IBM MQ group ID of the group or the message ID if only a single message is found. This variable is only set if you are monitoring for complete groups.
CurrentTimeStamp	A time stamp based on the local time that the monitor triggered at. The time stamp value is unique for the agent.
CurrentTimeStamp UTC	A time stamp based on the time, in the UTC time zone, that the monitor triggered at. The time stamp value is unique for the agent.

If the monitored resource is a directory

The following table shows the set of variable names that can be substituted in the task XML definition.

Table 19. Variables that can be substituted

Variable	Description
FilePath	The complete path name of the trigger file.
FileName	The file name part of the trigger.
LastModifiedTime	The time that the trigger file was last modified. This time is expressed as the local time of the time zone that the agent is running in and is formatted as an ISO 8601 time.
LastModifiedDate	The date that the trigger file was last modified. This date is expressed as the local date of the time zone that the agent is running in and is formatted as an ISO 8601 date.
LastModifiedTimeUTC	The time that the trigger file was last modified. This time is expressed as the local time converted to the UTC time zone and is formatted as an ISO 8601 time
LastModifiedDateUTC	The date that the trigger file was last modified. This date is expressed as the local date converted to the UTC time zone and is formatted as an ISO 8601 date.
AgentName	The name of the resource monitor agent.
CurrentTimeStamp	A time stamp that is based on the local time that the monitor triggered at. The time stamp value is unique for the agent.
CurrentTimeStampUTC	A time stamp that is based on the time in the UTC time zone that the monitor triggered at. The time stamp value is unique for the agent.

If the monitored resource is a trigger file

The following table shows the set of variable names that can be substituted when a resource monitor is using the contents of a trigger file to determine the files that need to be transferred.

Table 20. Variables that can be substituted when using a trigger file

Variable	Description
contentSource	The complete path name of the source file.

<i>Table 20. Variables that can be substituted when using a trigger file (continued)</i>	
Variable	Description
contentDestination	The complete path name of the destination file.

Variable names must be preceded by a dollar sign (\$) character and enclosed in braces, {}. For example, `${FilePath}` is replaced with the fully qualified file path of the matching trigger file.

There are two special keywords that can be applied to variable names to provide further refinement. These are:

token

The token index to substitute (starting at 1 from the left and starting at -1 from the right)

separator

A single character to tokenize the variable value. The default is the forward slash character (/) on UNIX platforms or backward slash character(\) on Windows platforms, but the separator can be any valid character that can appear in the variable value.

If the separator keyword is specified in a variable name, the variable value is split into tokens according to the separator character.

The value that is assigned to the token keyword is used as an index to select which token to use to replace the variable name. The token index is relative to the first character in the variable, and starts at 1. If the token keyword is not specified, the entire variable is inserted.

Any values that are substituted into an agent name in the message XML are treated in a not case-sensitive way. All Managed File Transfer Agent names are uppercase. If the value Paris is substituted into an agent attribute in the message XML, this value is interpreted as a reference to the agent PARIS.

Related concepts

[“Examples: Variable substitution for resource monitor definitions” on page 174](#)

Examples of variable substitution for resource monitor definitions using XML and IBM MQ Explorer.

Related reference

[“What to do if variable substitution causes multiple files to go to a single file name” on page 314](#)

For Managed File Transfer, if you are monitoring a directory and transferring multiple files from a source to a destination location and you are using `${FileName}` variable substitution, you must test the variable substitution results. The results need to be tested because the use of variable substitution might cause unexpected combinations of file transfer commands to be invoked.

Examples: Variable substitution for resource monitor definitions

Examples of variable substitution for resource monitor definitions using XML and IBM MQ Explorer.

Examples showing how variable substitution works

Assuming that the file path to the matching trigger file is

`c:\MONITOR\REPORTS\Paris\Report2009.doc` on Windows and `/MONITOR/REPORTS/Paris/Report2009.doc` on UNIX platforms, the variables are substituted as shown in the following table.

<i>Table 21. How the variables are substituted</i>	
Variable specification	After variable substitution
<code>\${FilePath}</code>	Windows <code>c:\MONITOR\REPORTS\Paris\Report2009.doc</code> UNIX <code>/MONITOR/REPORTS/Paris/Report2009.doc</code>

<i>Table 21. How the variables are substituted (continued)</i>	
Variable specification	After variable substitution
<code>\${FilePath{token=1}{separator=.}}</code>	Windows :c:\MONITOR\REPORTS\Paris\Report2009 UNIX :/MONITOR/REPORTS/Paris/Report2009
<code>\${FilePath{token=2}{separator=.}}</code>	Windows : doc UNIX : doc
<code>\${FilePath{token=3}}</code>	Windows : REPORTS UNIX : Paris

You can also specify a negative token index to select tokens relative to the last character of the variable, as shown in the following table. The examples in the table use the same variable value, `c:\MONITOR\REPORTS\Paris\Report2009.doc` on Windows and `/MONITOR/REPORTS/Paris/Report2009.doc` on UNIX platforms..

<i>Table 22. Examples of using a negative token index</i>	
Variable specification	After variable substitution
<code>\${FilePath}</code>	Windows :c:\MONITOR\REPORTS\Paris\Report2009.doc UNIX :/MONITOR/REPORTS/Paris/Report2009.doc
<code>\${FilePath{token=-2}{separator=.}}</code>	Windows :c:\MONITOR\REPORTS\Paris\Report2009 UNIX :/MONITOR/REPORTS/Paris/Report2009
<code>\${FilePath{token=-2}{separator=\\}}</code>	Windows : Paris UNIX : Paris
<code>\${FilePath{token=-4}}</code>	Windows : MONITOR UNIX : MONITOR

The variables that are used for substitution are only available for the following positive trigger conditions and the `noSizeChange` option, which is an exception to the positive trigger condition rule:

- `match`
- `fileSize`
- `noSizeChange`

If a `noMatch` condition is used, and there are substitution variable names in the task definition, the task is not called, and the monitor raises a return code of 110 and error message `BFGDM0060E`.

Example using XML

The following example task definition XML uses the monitor agent name as the source agent for the transfer (`Paris`), uses the penultimate directory name in the file path as the destination agent name for the transfer (`Report2009`), and renames the transferred file to be the root of the trigger file name with an extension of `.rpt`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
```

```

xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedTransfer>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <sourceAgent agent="{AgentName}" QMgr="QM1"/>
  <destinationAgent agent="{FilePath{token=-2}}" QMgr="QMD"/>
  <transferSet>
    <item mode="binary" checksumMethod="MD5">
      <source recursive="false" disposition="leave">
        <file>c:/incoming/reports/summary/report.doc</file>
      </source>
      <destination type="file" exist="overwrite">
        <file>/reports/{FileName{token=1}{separator=.}}.rpt</file>
      </destination>
    </item>
  </transferSet>
</managedTransfer>
</request>

```

This results in the task XML being transformed to:

```

<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="QM1"/>
    <destinationAgent agent="Paris" QMgr="QMD"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:/incoming/reports/summary/report.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/Report2009.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>

```

The variable `{FilePath{token=-2}}` in the `<destinationAgent>` element's `agent` attribute is replaced with the value `Paris`. This value is treated in a not case-sensitive way and interpreted as a reference to the agent `PARIS`.

Examples using IBM MQ Explorer

When creating a resource monitor through the IBM MQ Explorer, and once the monitor properties and trigger conditions have been specified, the option is given to add transfer items to the monitor. The following examples demonstrate how the `{FilePath}` and `{FileName}` variables can be used in the "**Add a transfer item panel**" to customize transfers resulting from a resource monitor match.

Example 1

In order to simply transfer the source file to another location when a trigger condition is met, the `{FilePath}` variable can be used:

- Set the source **File name** to be `{FilePath}`.
- From the dropdown menu of **Type** for the destination, select **Directory**.
- Set the destination **File name** to be the location to which you wish the source file to be transferred, for example, this could be `C:\MFT\out\`.

Example 2

In order to transfer the source file to another location and change the extension of the file, the `${FileName}` variable can be used in conjunction with the `${FilePath}` variable:

In the following example it is assumed that the file path of the source file is equal to `C:\MONITOR\REPORTS\Paris\Report2009.doc`:

- Set the source **File name** to be `${FilePath}`.
- Set the destination **File name** to be the location to which you wish the source file to be transferred, followed by `${FileName}{token=1}{separator=.}`, followed by the new extension of the file. For example, this could be `C:\MFT\out\${FileName}{token=1}{separator=.}.rpt`, which would equate to `C:\MFT\out\Report2009.rpt` with the source file name.

Example 3

In order to use part of the file path of the source file to determine the destination of the transfer, the `${FilePath}` variable can be used in conjunction with token and separator specifications.

In the following example it is assumed that the file path of the source file is equal to `C:\MONITOR\REPORTS\Paris\Report2009.doc`.

It is possible to use part of the source file path to determine the destination of the file. Using the file path example of `C:\MONITOR\REPORTS\Paris\Report2009.doc`, if the file were to be transferred to a folder depending upon the location of the source file, that is, `Paris` in this example, then the following could be done:

- Set the source **File name** to be `${FilePath}`.
- Set the destination **File name** to be the destination to where the folders for each location are situated, and then append the destination part of the file path and the file name. For example, this could be `C:\MFT\out\${FilePath}{token=-2}{separator=} \ ${FileName}`, which would equate to `C:\MFT\out\Paris\Report2009.doc` with the source file name.

Related concepts

[“Customizing MFT resource monitor tasks with variable substitution” on page 172](#)

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Related reference

[“What to do if variable substitution causes multiple files to go to a single file name” on page 314](#)

For Managed File Transfer, if you are monitoring a directory and transferring multiple files from a source to a destination location and you are using `${FileName}` variable substitution, you must test the variable substitution results. The results need to be tested because the use of variable substitution might cause unexpected combinations of file transfer commands to be invoked.

Monitoring a queue and using variable substitution

You can monitor a queue and transfer messages from the monitored queue to a file by using the **ftCreateMonitor** command. The value of any IBM MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called `AGENT_VENUS`, which connects to `QM_VENUS`. The queue that `AGENT_VENUS` monitors is called `START_QUEUE` and is located on `QM_VENUS`. The agent polls the queue every 30 minutes.

When a complete group of messages is written to the queue the monitor task sends the group of messages to a file at one of a number of destination agents, all of which connect to the queue manager QM_MARS. The name of the file that the group of messages is transferred to is defined by the IBM MQ message property `usr.fileName` on the first message in the group. The name of the agent that the group of messages is sent to is defined by the IBM MQ message property `usr.toAgent` on the first message in the group. If the `usr.toAgent` header is not set, the default value to be used for the destination agent is AGENT_MAGENTA.

When you specify `useGroups="true"`, if you do not also specify `groupId="{GROUPID}"`, the transfer just takes the first message on the queue. For example, if you are using variable substitution to generate the `fileName`, it is therefore possible that the contents of a `.txt` will not be correct. This is because the `fileName` is generated by the monitor, but the transfer actually gets a message that is not the one that should generate the file called `fileName`.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS"/>
    <destinationAgent agent="{toAgent}" QMgr="QM_MARS"/>
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <queue useGroups="true" groupId="{GROUPID}">START_QUEUE</queue>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/{fileName}.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The variables that are replaced with the values of IBM MQ message headers are highlighted in **bold**. This task XML is saved to the file `/home/USER1/task.xml`

2. Create a resource monitor to monitor the queue START_QUEUE.

Submit the following command:

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. A user or program writes a group of messages to the queue START_QUEUE.

The first message in this group has the following IBM MQ message properties set:

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. The monitor is triggered when the complete group is written. The agent substitutes the IBM MQ message properties into the task XML.

This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
```

```

<managedTransfer>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS"/>
  <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS"/>
  <transferSet>
    <item mode="binary" checksumMethod="none">
      <source>
        <queue useGroups="true" groupId="{GROUPID}">START_QUEUE</queue>
      </source>
      <destination type="file" exist="overwrite">
        <file>/reports/larmer.rpt</file>
      </destination>
    </item>
  </transferSet>
</managedTransfer>
</request>

```

Results

The transfer that is defined by the task XML is performed. The complete group of messages that are read from the START_QUEUE by AGENT_VENUS is written to a file called /reports/larmer.rpt on the system where AGENT_VIOLET is running.

What to do next

Transferring each message to a separate file

If you want to monitor a queue and have every message transferred to a separate file, you can use a similar technique to the one described previously in this topic.

1. Create the monitor as described previously, specifying the **-tr completeGroups** parameter on the **fteCreateMonitor** command.
2. In the task XML specify the following:

```

<queue useGroups="true" groupId="{GROUPID}">START_QUEUE</queue>

```

However, when you put the messages onto the source queue, do not put them in an IBM MQ group. Add IBM MQ message properties to each message. For example, specify the `usr.filename` property with a unique file name value for each message. This effectively causes the Managed File Transfer Agent to treat each message on the source queue as a separate group.

Monitor retry behavior for message-to-file transfers

If a message-to-file transfer that is triggered by a resource monitor fails and leaves the message group that triggered the monitor on the queue, that transfer is resubmitted at subsequent poll intervals. The number of times that the transfer is resubmitted is limited by the **monitorGroupRetryLimit** property of the monitoring agent.

The number of times that the message-to-file transfer has been triggered is determined from the MQMD backout count of the first message in the group.

Each time a new message-to-file transfer is triggered a new transfer ID is generated for the transfer task.

If the agent is restarted the monitor triggers a transfer again even if the number of times the transfer has been triggered has exceeded the value of **monitorGroupRetryLimit**. If this transfer attempt causes the number of times that the transfer has been triggered to exceed the value of **monitorGroupRetryLimit**, the agent writes an error to its event log.

A single message is treated as if it was a single group, and the transfer is triggered again at each poll interval while the message remains on the queue and while the number of times the transfer has been triggered is less than the value of **monitorGroupRetryLimit**.

Setting the `monitorGroupRetryLimit` property

The value of the `monitorGroupRetryLimit` property is the maximum number of times that a monitor triggers a message-to-file transfer again if the message group still exists on the queue. The default value of this property is 10. The value of this property can be set to any positive integer value or -1. If the value -1 is specified for this property, the monitor triggers the transfer again an unlimited number of times, until the trigger condition is not satisfied.

To set the `monitorGroupRetryLimit` property on the monitoring agent, perform the following steps:

1. Stop the monitoring agent, using the **`fteStopAgent`** command.
2. Edit the monitoring agent `agent.properties` file to include the line `monitorGroupRetryLimit=number_of_retries`. The `agent.properties` file is located in the directory `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/monitoring_agent_name`.
3. Start the monitoring agent, using the **`fteStartAgent`** command.

Related tasks

[“Example: Configuring an MFT resource” on page 171](#)

You can specify an IBM MQ queue as the resource to be monitored by a resource monitor by using the **`-mq`** parameter with the **`fteCreateMonitor`** command.

Related reference

[“The MFT `agent.properties` file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Using a trigger file

You can use the contents of a trigger file in a resource monitor to define a set of files to transfer in a single transfer request. Each time a matching trigger file is detected, its contents are parsed for source file paths and optionally for destination file paths. These file paths are then used to define file items in the task transfer XML file that you specify, which is submitted as single transfer request to the agent. The definition of the resource monitor determines whether trigger content is enabled.

You can enable file content triggering when you create a monitor by specifying the **`-tc`** (trigger content) parameter. This **`-tc`** parameter applies only to the file trigger options `match` and `noSizeChange`. For more information about creating a monitor, see [“`fteCreateMonitor`: create an MFT resource monitor” on page 394](#).

When using a trigger content file, the default format of each line is either:

- A single source file path, or
- A source file path and a destination file path, separated by a comma

where white space characters are handled as part of the file paths. It is possible to change the default line format by specifying the **`-tcr`** and **`-tcc`** parameters on the **`fteCreateMonitor`** command. For more information, see [“Advanced options” on page 181](#).

After a trigger file is parsed, a list of file paths is generated and applied to the transfer task XML that you specified. As with all monitors, the format of the transfer task XML is a complete transfer task XML generated by the **`fteCreateTransfer`** command with a single item or file defined. The single item must use the substitution variables `${contentSource}`, and optionally `${contentDestination}`, as replacements for the source and destination file paths. The monitor expands the transfer task XML to include a file item for each line (file path) in the trigger file.

You cannot use file content triggering with the **`-bs`** parameter because the **`-tc`** parameter implies one transfer request for each trigger file.

Example

The following example defines a monitor to trigger on a file that ends in `trig` and reads the file paths in that file.

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${contentSource}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

The **fteCreateTransfer** command creates a file that is called `task.xml` for a single file with a source file path of `${sourceContent}`. For example:

```
<item checksumMethod="MD5" mode="binary">
  <source disposition="leave" recursive="false">
    <file>${contentSource}</file>
  </source>
</item>
```

The **fteCreateMonitor** command scans for files that end in `trig` in the `/home/trigdir` directory and uses the contents to create a single transfer request that is based on the `task.xml` for all paths in that trigger file. The format of the trigger file must be one file path (source only) on each line with no comma separator. For example:

```
/home/file/first.txt
/home/file/second.txt
/home/different/third.txt
:
```

All files are delivered to the `/file/destdir` directory with its file name and not its file path, that is, `/home/file/first.txt` is delivered to `/file/destdir/first.txt`.

Alternatively, if you change the **-dd /file/destdir** parameter in the **fteCreateTransfer** command to **-df \${contentDestination}** and the format of the content of a trigger file to *source file path,destination file path*, you can define different destination paths for the same destination agent. For example:

```
/home/file/first.txt,/home/other/sixth.txt
```

The destination location then becomes `/home/other/sixth.txt`.

The substitution variables can be tokenized. For example, you can separate the file name part from the provided path using `${contentDestination{token=-1}}`. Therefore, if the **fteCreateTransfer** destination is defined as **-df /file/destdir/\${contentDestination{token=-1}}**, the new destination for `/home/file/first.txt` is `/file/destdir/sixth.txt`.

Advanced options

You can change the default line format for the content of the trigger file by using the **-tcr regex** parameter. Supply a regular expression that matches the required line format and supplies either one or two capture groups. The first capture group is the source and the second, optional, capture group is the destination. For example:

- The source and destination path are separated by a hyphen:

```
((?:[^\-])+)-((?:[^\-])?)
```

In this example, the separator is defined in three locations and all three instances of the hyphen, `-`, can be changed to any character. Ensure that you escape any special characters.

- The source and destination paths are separated by a comma with trailing spaces. Comments that are indicated by a number sign (`#`) are ignored.

```
((?:[^\,])+),((?:[^\,])+) *(?:#\.*)+
```

File paths cannot contain the number sign (#). Typically an entry is as follows: /home/source/from.txt,/home/destination/to.txt # *some comment*.

If you use the **-tcx** parameter, ensure that the regular expression is well designed and tested so that the expression can detect errors and correctly parse the trigger files.

You can reverse the order of the capture by using the **-tcc destSrc** parameter. If you specify this parameter, the first capture group is the destination file path and the second group is the source file path.

How errors are handled

Empty trigger file

If the trigger file is empty, the outcome is no file transfer. That is, the monitor creates a transfer request but no file items are specified.

Trigger file with errors

If any entry in a trigger file fails to parse against the expected format, no transfer request is generated. A monitor error log is published and the error is also logged in the event log. The trigger file is marked as processed and the monitor does not attempt to process the file again until the file has been updated.

Mismatching transfer task XML

The transfer task XML must match the trigger file, that is if the transfer task XML has both `${sourceContent}` and `${destinationContent}`, all trigger files for that monitor must have source and destination file paths and similarly for the reverse. In the first case the monitor reports a substitution failure of the `${destinationContent}` if the trigger file supplies the source file path only.

Examples

The following example is a basic content trigger where the contents of a trigger file has a source file path only:

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -dd /file/destdir ${sourceContent}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
```

The **-tcx** parameter defines two capture groups of a sequence of any characters that are separated by a space character. The **-tcc destSrc** parameter and option indicate that the capture groups are to be processed as destination then source.

```
fteCreateTransfer -gt task.xml -sa SrcAgent -da DestAgent -df ${destinationContent} ${
sourceContent}
fteCreateMonitor -mn TrigMonitor -md /home/trigdir -mt task.xml -ma SrcAgent -tr "match,*.trig"
-tc
-tcx "((?:[^\ ])+) ((?:[^\ ])+)" -tcc destSrc
```

V 9.0.3 Logging MFT resource monitors

From IBM MQ 9.0.3, Managed File Transfer includes an improved method for obtaining diagnostic information about resource monitors.

About this task

From IBM MQ 9.0.3, you can use logging for resource monitors. You can either use the **fteSetAgentLogLevel** command or the `agent.properties` file to control resource monitor logging.

Note that existing trace points are still used for capturing information.

The resource monitor logs are written to a file named `resmoneventN.log`, where *N* stands for a number; for example, `resmonevent0.log`.



Attention: All resource monitors of an agent write to the same log file.

For example output of a `resmoneventN.log` file, see [“What to do if your MFT directory resource monitor is not triggering files”](#) on page 300.

The following table lists the type of events the resource monitor writes to the log file. The third column describes the log level needed to capture each event where the lowest level is INFO and highest is VERBOSE.

Note that setting a higher log level, also writes lower level events. For example, setting log level to MODERATE also writes INFO level events, but not VERBOSE level events.

Number	Event	Log level	Description
1	Monitor create	INFO	A resource monitor has been created.
2	Monitor delete	INFO	A resource monitor has been deleted.
3	Monitor stop	INFO	A resource monitor has been stopped.
4	Monitor start	INFO	A resource monitor has been started.
5	Monitor start poll	INFO	A resource monitor started a new poll cycle.
6	Monitor end poll	INFO	A resource monitor poll cycle has ended.
7	Pattern match	VERBOSE	A file on the trigger monitor directory, or a message in a queue that matches the specified pattern, has been found.
8	Pattern mismatch	VERBOSE	An unmatched file on the trigger monitor directory, or a message in a queue that does not match the specified pattern, has been found.
9	Transfer submit	INFO	A transfer has been initiated by the resource monitor.
10	Directory too deep	VERBOSE	The directory monitored by the resource monitor contains more sub-directories to poll, than the number specified in the resource monitor configuration.
11	File locked	MODERATE	The trigger file monitored by the resource monitor is locked by another process.
12	File size small	MODERATE	The trigger file is smaller than the size specified in the resource monitor configuration.
13	File size unstable	MODERATE	The trigger file is being changed more frequently than expected by the resource monitor configuration.
14	Too many polls	MODERATE	A resource monitor has polled an unstable trigger file too many times.
15	Items matched	INFO	Total number of trigger files found in the directory polled by a resource monitor.
16	Items in transfer	INFO	Total number of items in the transfer request.

Number	Event	Log level	Description
17	FDC or exception	MODERATE	A resource monitor has generated an exception.
18	Transfer request	INFO	Transfer request submitted by resource monitor.
19	Monitor start failed	MODERATE	A resource monitor failed to start.

Procedure

- To use the **fteSetAgentLogLevel** to turn resource monitor logging on and off, see [fteSetAgentLogLevel](#) for a description of the **logMonitor** parameter, and examples of how you use the different options.
- To use the `agent.properties` file to control resource monitor logging, see “The MFT `agent.properties` file” on page 526 for a description of the additional properties that allow you to carry out the following logging activities:
 - Turn logging on or off
 - Limit the size of each log file
 - Limit the number of logs that resource monitors can generate

Example

The following sample message sets verbose level logging for agent HA2, on queue manager MFTDEMO:

```
<?xml version="1.0"?>
<log:log version="6.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:log="https://www.ibm.com/log">
  <log:originator>
    <log:request>
      <log:hostname>192.168.7.1</log:hostname>
      <log:userid>johndoe</log:userid>
    </log:request>
  </log:originator>
  <log:endpoint agent="HA2" QMgr="MFTDEMO"/>
  <log:logMonitor>MON1="verbose"</log:logMonitor>
</log:log>
```

Related reference

[fteSetAgentLogLevel](#) command

Use the **fteSetAgentLogLevel** command to enable or disable diagnostic information logging for file transfers between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers.

“The MFT `agent.properties` file” on page 526

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Backing up and restoring MFT resource monitors

You can back up the resource monitors that you want to have available for future use by exporting their definitions to an XML file that you can then import to create a new resource monitor from the backup.

About this task

You might need to back up the resource monitors that you have previously defined so that you can re-use their definitions in the future, for example to re-create the resource monitors in a different infrastructure or if a resource monitor needs to be re-created because of queue manager issues.

You can back up a single resource manager definition by using either the **fteCreateMonitor** command or the **fteListMonitors** command with the **-ox** parameter. In both cases, the resource manager definition is backed up by exporting it to an XML file. You can then use the **-ix** parameter of the **fteCreateMonitor** command to create a new resource manager by importing the definition from the XML file.

With the **-ox** parameter, you can back up only one resource monitor definition at a time.

V 9.0.5 From IBM MQ 9.0.5, the **-od** parameter is added to the **fteListMonitors** command. By specifying this parameter, you can back up more than one resource monitor at a time by exporting their definitions in bulk to a specified directory. Each resource monitor definition is saved to a separate XML file with a name in the format *agent name.monitor name.xml*.

V 9.0.5 The **-od** parameter is particularly useful if you have a large number of resource monitors that you want to back up because you need to run the **fteListMonitors -od** command once only, instead of having to run the **fteListMonitors -ox** command separately for each resource definition, or use a separate script to run the **fteListMonitors -ox** command for each resource monitor.

Procedure

- To back up the definition of one resource monitor by exporting it to an XML file, use either of the following commands:
 - The **fteCreateMonitor** command with the **-ox** parameter.
 - The **fteListMonitors** command with the **-ox** parameter.

When you are using the **-ox** parameter, you must also specify the **-ma** and **-mn** parameters, as shown in the following example:

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
```

- **V 9.0.5**
To back up multiple resource monitor definitions by exporting them to XML files in a specified directory, use the **fteListMonitors** command with the **-od** parameter as shown in the following example:

```
fteListMonitors -od /usr/mft/resmonbackup
```

You must specify a valid target directory when you are backing up resource monitors in bulk. Not specifying a target path results in an error message as shown in the following example:

```
BFGCL0762E: Output directory not specified. Rerun the command specifying a valid path.
```

The **-od** parameter must not be combined with the **-ox** parameter, otherwise the following error message is displayed:

```
BFGCL0761E: It is not valid to specify both the '-od' and '-ox' parameters together.
```

You can define a particular set of resource monitors to include in the backup. For instance, by using the **-ma** parameter to specify the name of an agent, you can back up all the resource monitors for that agent, as shown in the following example:

```
fteListMonitors -ma AGENT1 -od /usr/mft/resmonbackup
```

You can also use wildcard matching by including an asterisk character (*) character when defining a pattern to use for matching agent names or monitor names, or both. The following example backs up all the resource monitors that have names matching a specified pattern and that are in an agent with a name matching a specified pattern:

```
fteListMonitors -ma AGENT* -mn MON* -od /usr/mft/resmonbackup
```

While the command is running, it displays the following progress report messages:

A total of *number* matching resource monitor definitions found.
index of *number* resource monitor definitions saved to file system.

If you are using the verbose option, the running total is still displayed, but instead of displaying *index* of *number* resource monitor definitions saved to file system

the command displays the name of the monitor definition being saved, for example:

```
BFGCL0762I: Definition of monitor 'FILEMON' of agent 'XFERAGENT' saved as  
FILEMON.XFERAGENT.XML to file system.
```

- **V 9.0.5**

To back up one resource monitor for a particular agent by exporting it to an XML file in a specified directory, use the **fteListMonitors** command with the **-od** parameter:

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -od /usr/mft/resmonbackup
```

Using the **-od** parameter to back up a single resource monitor is similar to using the **-ox** parameter, except that the output file name is in the format *agent name.monitor name.xml*.

- To restore resource monitor definitions from a backup, use the **fteCreateMonitor** command with the **-ix** parameter as shown in the following example:

```
fteCreateMonitor -ix file name
```

For more examples of how to use the **-od** parameter, see [“fteListMonitors: list MFT resource monitors”](#) on page 455.

Related reference

[“fteCreateMonitor: create an MFT resource monitor”](#) on page 394

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

[“fteListMonitors: list MFT resource monitors”](#) on page 455

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

Working with file transfer templates

You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the IBM MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your Managed File Transfer network.

About this task

To create a transfer template from the command line, use the [fteCreateTemplate](#) command. Then when you want to submit a transfer template that you created on the command line, click **Submit** in IBM MQ Explorer.

To view transfer templates in the IBM MQ Explorer, use the following steps:

Procedure

1. Expand **Managed File Transfer** in the Navigator view. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are listed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.

3. Click **Transfer Templates**. The **Transfer Templates** window is displayed in the Content view.
4. The **Transfer Templates** window lists the following details about your file transfers:
 - a) **Name** The name of your file transfer template.
 - b) **Source** The name of the agent used to transfer the file from the source system.
 - c) **Source File** The name of the file to be transferred on its host system.
Expand the transfer template information to view this field.
 - d) **Destination** The name of the agent used to receive the file at the destination system.
 - e) **Destination File** The name of the file after it is transferred to the destination system.
Expand the transfer template information to view this field.
 - f) **Scheduled Start (selected time zone)** The time and date that the file transfer is scheduled to start in the time zone used by the administrator. To change the time zone displayed, click **Window > Preferences > IBM MQ Explorer > Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.
 - g) **Trigger Events** The type of event that triggers the file transfer to start. The type can be one of the following values: `exists`, `does not exist`, or `exceeds`.

Results

To refresh what is displayed in the **Transfer Templates** window, click the Refresh button  on the Content view toolbar.

To submit a transfer template and start the transfer defined in the template, right-click the template name and click **Submit**.

To change a transfer template, right-click the template name and click **Edit**. All files included in the original template are listed as part of a transfer group, even if they were not included as part of a group in the original template. If you want to remove a file from the template you must select the file specification from the group and click **Remove selected**. If you want to add new file specifications to the template use the fields in the template panel and click the **Add to group** button. When you have made your edits, you are prompted to give the edited template a new name.

To create a file transfer from a transfer template, right-click the template name and click **Edit as New Transfer**.

To create a duplicate copy of a transfer template, right-click the template name and click **Duplicate**. The duplicate transfer template is automatically saved with the same name as the original template, appended with "(copy)".

To delete a transfer template, right-click the template name and click **Delete**.

Related tasks

[“Creating a file transfer template using IBM MQ Explorer” on page 188](#)

You can create a file transfer template from IBM MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference

[“fteCreateTemplate: create new file transfer template” on page 401](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[“fteListTemplates” on page 459](#)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

[“fteDeleteTemplates” on page 449](#)

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

Creating a file transfer template using IBM MQ Explorer

You can create a file transfer template from IBM MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

About this task

To create a file transfer template from the command line, use the [fteCreateTemplate](#) command.

To create a file transfer template using the **Create New Template for Managed File Transfer** wizard in IBM MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Start the **Create New Template for Managed File Transfer** wizard by right-clicking **Transfer Templates** and then clicking **New Template**.
4. Follow the instructions on the wizard panels. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press Ctrl+F1 or Shift+F1.

If you have created a template that contains all the required transfer details, ensure that you select the **Save transfer settings as a template** check box on the **Transfer Summary** page if this check box is not already selected. Also enter a name for the template in the Name field. If you create a template that does not yet contain all of the required transfer details, the **Save transfer settings as a template** check box is automatically checked for you.

Related tasks

[“Working with file transfer templates” on page 186](#)

You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the IBM MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your Managed File Transfer network.

Related reference

[“fteCreateTemplate: create new file transfer template” on page 401](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[“fteListTemplates” on page 459](#)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

[“fteDeleteTemplates” on page 449](#)

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

Transferring data from files to messages

You can use the file-to-message feature of Managed File Transfer to transfer data from a file to a single message, or multiple messages, on an IBM MQ queue.

To perform file-to-message and message-to-file transfers both the source and destination agent of the transfer must either be at IBM WebSphere MQ 7.5 , or later, or at IBM WebSphere MQ File Transfer Edition 7.0.3, or later. For information about message-to-file transfers, see [“Transferring data from messages to files” on page 197](#).

The destination agent for a file-to-message transfer cannot be a protocol bridge agent or a Connect:Direct bridge agent.

You can transfer file data to IBM MQ message data. The IBM MQ messages can be read and used by applications. The following types of file-to-message transfer are supported:

- From a single file to a single message. The message does not have an IBM MQ group ID set.
- From a single file to multiple messages, by splitting the file into messages of a given length. The messages all have the same IBM MQ group ID.
- From a single file to multiple messages, by splitting a text file at a Java regular expression delimiter. The messages all have the same IBM MQ group ID.
- From a single file to multiple messages, by splitting a binary file at a hexadecimal delimiter. The messages all have the same IBM MQ group ID.

If you want to split a binary file using a sequence of bytes as the delimiter, use the **-sqdb** parameter of the **fteCreateTransfer** command. For more information, see [-sqdb parameter](#).

By default the messages created by a file-to-message transfer are persistent. The messages can be set to be non-persistent or to have the persistence value defined by the destination queue.

If you specify that a file is split into multiple messages, all messages created from the file have the same IBM MQ group ID. If you do not specify that a file is split into multiple messages, only one message is created from the file and this message does not have the IBM MQ group ID set.

If you are transferring files to large messages, or many small messages, you might need to change some IBM MQ or Managed File Transfer properties. For information about, see [“Guidance for setting MQ attributes and MFT properties associated with message size” on page 306](#).

Note: If the destination queue is either a clustered queue, or an alias to a clustered queue, you will get an error message when transferring a file into a queue if the agent property `enableClusterQueueInputOutput` has not been set to true. For more information see [“What to do if the destination queue is a clustered queue, or an alias to a clustered queue” on page 303](#)

Configuring an agent to perform file-to-message transfers

By default agents cannot perform file-to-message or message-to-file transfers. To enable this function you must set the agent property `enableQueueInputOutput` to true. To enable writing to IBM MQ clustered queues, you must also set the agent property `enableClusterQueueInputOutput` to true.

About this task

If you attempt to perform a file-to-message transfer to a destination agent that does not have the `enableQueueInputOutput` property set to true, the transfer fails. The transfer log message that is published to the coordination queue manager contains the following message:

```
BFGI00197E: An attempt to write to a queue was rejected by the destination agent. The agent must have enableQueueInputOutput=true set in the agent.properties file to support transferring to a queue.
```

To enable the agent to write to and read from queues perform the following steps:

Procedure

1. Stop the destination agent using the **fteStopAgent** command.
2. Edit the agent `.properties` file to include the line `enableQueueInputOutput=true`.
The agent `.properties` file is located in the directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/destination_agent_name`.
3. Optional: Edit the agent `.properties` file to include the line `enableClusterQueueInputOutput=true`. The agent `.properties` file is located in the directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/destination_agent_name`.
4. Start the destination agent using the **fteStartAgent** command.

Example: Transferring a single file to a single message

You can specify a queue as the destination of a file transfer by using the **-dq** parameter with the **fteCreateTransfer** command. The source file must be smaller than the maximum message length set on the destination queue. The destination queue does not have to be on the same queue manager as the queue manager that the destination agent connects to, but these two queue managers must be able to communicate.

About this task

The source file is called `/tmp/single_record.txt` and is located on the same system as the source agent, AGENT_NEPTUNE. The source agent, AGENT_NEPTUNE, uses the queue manager QM_NEPTUNE. The destination agent is AGENT_VENUS and this agent connects to the queue manager QM_VENUS. The destination queue, RECEIVING_QUEUE, is located on the queue manager QM_MERCURY. QM_MERCURY is in the same IBM MQ network as, and can be accessed by, the queue manager QM_VENUS.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -dm QM_VENUS  
-dq RECEIVING_QUEUE@QM_MERCURY /tmp/single_record.txt
```

If the destination queue is on a different queue manager to the queue manager used by the destination agent you must specify the value of the **-dq** parameter in the following format `queue_name@queue_manager_name`. If you do not specify `@queue_manager_name` in the value, the destination agent assumes that the destination queue is located on the destination agent queue manager. The exception is when the `enableClusterQueueInputOutput` agent property has been set to true. In this case the destination agent will use standard IBM MQ resolution procedures to determine where the queue is located.

The source agent, AGENT_NEPTUNE, reads the data from the file `/tmp/single_record.txt` and transfers this data to the destination agent, AGENT_VENUS. The destination agent, AGENT_VENUS, sends the data to a persistent message on the queue `RECEIVING_QUEUE@QM_MERCURY`. The message does not have an IBM MQ group ID set.

Example: Splitting a single file into multiple messages by length

You can split a file into multiple IBM MQ messages by using the **-qs** parameter of the **fteCreateTransfer** command. The file is split into fixed-length sections, each of which is written to an individual message.

About this task

The source file is called `/tmp/source.file` and is 36 KB in size. The source file is located on the same system as the source agent `AGENT_NEPTUNE`. The source agent, `AGENT_NEPTUNE`, connects to the queue manager `QM_NEPTUNE`. The destination agent is `AGENT_MERCURY`, which connects to the queue manager `QM_MERCURY`. The destination queue, `RECEIVING_QUEUE`, is also located on the queue manager `QM_MERCURY`. The transfer splits the source file into sections that are 1 KB in size and writes each of these sections to a message on `RECEIVING_QUEUE`.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY  
-dq RECEIVING_QUEUE -qs 1K /tmp/source.file
```

The source agent, `AGENT_NEPTUNE`, reads the data from the file `/tmp/source.file` and transfers this data to the destination agent, `AGENT_MERCURY`. The destination agent, `AGENT_MERCURY`, writes the data to thirty-six 1 KB persistent messages on the queue `RECEIVING_QUEUE@QM_MERCURY`. These messages all have the same IBM MQ group ID and the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set.

Example: Splitting a text file into multiple messages using a regular expression delimiter

Transfer a single text file to multiple messages by splitting the file at each match of a given Java regular expression. To do this you use the **-dqdt** parameter of the **fteCreateTransfer** command.

About this task

The file is split into variable-length sections, each of which is written to an individual message. The text file is split at each point where the text in the file matches a given regular expression. The source file is called `/tmp/names.text` and has the following contents:

```
Jenny Jones,John Smith,Jane Brown
```

The regular expression that specifies where to split the file is the comma character (`,`).

The source file is located on the same system as the source agent `AGENT_NEPTUNE`, which connects to the queue manager `QM_NEPTUNE`. The destination queue, `RECEIVING_QUEUE`, is located on the queue manager `QM_MERCURY`. `QM_MERCURY` is also the queue manager used by the destination agent `AGENT_MERCURY`. The transfer splits the source file into sections and writes each of these sections to a message on `RECEIVING_QUEUE`.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY  
-dq RECEIVING_QUEUE -t text -dqdp postfix -dqdt "," /tmp/names.text
```

The source agent, `AGENT_NEPTUNE`, reads the data from the file `/tmp/names.text` and transfers this data to the destination agent, `AGENT_MERCURY`. The destination agent, `AGENT_MERCURY`, writes the

data to three persistent messages on the queue *RECEIVING_QUEUE*. These messages all have the same IBM MQ group ID and the last message in the group has the IBM MQ *LAST_MSG_IN_GROUP* flag set.

The data in the messages is as follows.

- First message:

```
Jenny Jones
```

- Second message:

```
John Smith
```

- Third message:

```
Jane Brown
```

Example: Splitting a text file with a regular expression delimiter and including the delimiter in the messages

Transfer a single text file to multiple messages by splitting the file at each match of a given Java regular expression and include the regular expression match in the resulting messages. To do this you use the **-dqdt** and **-qi** parameters of the **fteCreateTransfer** command.

About this task

Transfer a single text file to multiple messages on a queue. The file is split into variable-length sections, each of which is written to an individual message. The text file is split at each point where the text in the file matches a given regular expression. The source file is called `/tmp/customers.text` and has the following contents:

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314

Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42

Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

The regular expression that specifies where to split the file is `Customer\number:\s\d+`, which matches the text "Customer number: " followed by any number of digits. Regular expressions specified at the command line must be enclosed in double quotation marks to prevent the command shell evaluating the regular expression. The regular expression is evaluated as a Java regular expression. For more information, see [“Regular expressions used by MFT”](#) on page 633.

By default the number of characters that a regular expression can match is set to five. The regular expression used in this example matches strings that are longer than five characters. To enable matches that are longer than five characters edit the agent properties file to include the property **maxDelimiterMatchLength**.

By default, the text that matches the regular expression is not included in the messages. To include the text that matches the regular expression in the messages, as in this example, use the **-qi** parameter. The source file is located on the same system as the source agent `AGENT_NEPTUNE`, which connects to the queue manager `QM_NEPTUNE`. The destination queue, `RECEIVING_QUEUE`, is located on the queue manager `QM_MERCURY`. `QM_MERCURY` is also the queue manager used by the destination agent `AGENT_MERCURY`. The transfer splits the source file into sections and writes each of these sections to a message on `RECEIVING_QUEUE`.

Procedure

1. Stop the destination agent using the following command:

```
fteStopAgent AGENT_MERCURY
```

2. Add the following line to the agent properties file for AGENT_MERCURY:

```
maxDelimiterMatchLength=25
```

Note: Increasing the value of **maxDelimiterMatchLength** can decrease performance.

3. Start the destination agent using the following command:

```
fteStartAgent AGENT_MERCURY
```

4. Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY  
-dq RECEIVING_QUEUE  
text -dqdt "Customer\snumber:\s\d+" -qi -dqdp postfix /tmp/customers.text
```

The source agent, AGENT_NEPTUNE, reads the data from the file `/tmp/customers.text` and transfers this data to the destination agent, AGENT_MERCURY. The destination agent, AGENT_MERCURY, writes the data to three persistent messages on the queue RECEIVING_QUEUE. These messages all have the same IBM MQ group ID and the last message in the group has the IBM MQ LAST_MSG_IN_GROUP flag set.

The data in the messages is as follows.

- First message:

```
Customer name: John Smith  
Customer contact details: john@example.net  
Customer number: 314
```

- Second message:

```
Customer name: Jane Brown  
Customer contact details: jane@example.com  
Customer number: 42
```

- Third message:

```
Customer name: James Jones  
Customer contact details: jjones@example.net  
Customer number: 26
```

Example: Setting IBM MQ message properties on a file-to-message transfer

You can use the **-qmp** parameter on the **fteCreateTransfer** command to specify whether IBM MQ message properties are set on the first message written to the destination queue by the transfer. IBM MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing IBM MQ Message Descriptor (MQMD) or MQRFH2 headers.

About this task

Include the parameter **-qmp true** in the **fteCreateTransfer** command. In this example, the MQMD user ID of the user submitting the command is `larmer`.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM -qmp true
-t text /tmp/source_file.txt
```

The IBM MQ message properties of the first message written by the destination agent, AGENT_SATURN, to the queue, MY_QUEUE, on queue manager, MyQM, are set to these values:

```
usr.WMQFTETransferId=414cbaedefa234889d999a8ed09782395ea213ebbc9377cd
usr.WMQFTETransferMode=text
usr.WMQFTESourceAgent=AGENT_JUPITER
usr.WMQFTEDestinationAgent=AGENT_SATURN
usr.WMQFTEFileName=source_file.txt
usr.WMQFTEFileSize=1024
usr.WMQFTEFileLastModified=1273740879040
usr.WMQFTEFileIndex=0
usr.WMQFTEMqmdUser=larmer
```

Example: Setting user-defined properties on a file-to-message transfer

User-defined metadata is set as an IBM MQ message property on the first message written to the destination queue by the transfer. IBM MQ message properties enable an application to select messages to process, or to retrieve information about a message without accessing IBM MQ Message Descriptor (MQMD) or MQRFH2 headers.

About this task

Include the parameters `-qmp true` and `-md account=123456` in the **fteCreateTransfer** command, to set the `usr.account` property to 123456 in the RFH2 header.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM
-qmp true -md account=123456 /tmp/source_file.txt
```

In addition to the standard set of IBM MQ message properties, the user-defined property is set in the message header of the first message written by the destination agent, AGENT_SATURN, to the queue, MY_QUEUE, on queue manager, MyQM. The header is set to the following value:

```
usr.account=123456
```

The prefix `usr` is added to the beginning of the name of the user-defined metadata.

Example: adding a user-defined message property for a file-to-message transfer

If you are using Managed File Transfer for message-to-file managed transfers, you can include a user-defined message property for the resulting message.

About this task

You can use any of the following methods to define a custom message property:

- Specify the **-md** parameter on the transfer request. For more information, see [“Example: Setting user-defined properties on a file-to-message transfer”](#) on page 194.

- Use an Ant task; you can use either `fte:filecopy` or `fte:filemove`. The following example is a `fte:filecopy` task:

```
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">
<!-- Initialise the properties used in this script.-->

<target name="init" description="initialise task properties">
  <property name="src.file" value="/home/user/file1.bin"/>
  <property name="dst.queue" value="TEST.QUEUE@qm2"/>
  <fte:uuid property="job.name" length="8"
prefix="copyjob#"/>
</target>
<target name="step1" depends="init" description="transfer file">

<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">

<fte:metadata>
<fte:entry name="fileName" value="{FileName}"/>
</fte:metadata>

<fte:filespec srcfilespec="{src.file}" dstqueue="{dst.queue}"
dstmsgprops="true"/>

</fte:filecopy>

</target>
</project>
```

- Use a resource monitor and variable substitution. The following example shows some transfer task XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor
xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="5.00"
xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./Monitor.xsd">
  <name>METADATA</name>
  <pollInterval units="minutes">5</pollInterval>
  <batch maxSize="5"/>
  <agent>AGENT1</agent>
  <resources>
    <directory recursionLevel="0">e:\temp</directory>
  </resources>
  <triggerMatch>
    <conditions>
      <allOf>
        <condition>
          <fileMatch>
            <pattern>*.txt</pattern>
          </fileMatch>
        </condition>
      </allOf>
    </conditions>
  </triggerMatch>
  <tasks>
    <task>
      <name/>
      <transfer>
        <request version="5.00"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:namespaceSchemaLocation="FileTransfer.xsd">
          <managedTransfer>
            <originator>
              <hostName>mqjason.raleigh.ibm.com.</hostName>
              <userID>administrator</userID>
            </originator>
            <sourceAgent QMgr="AGENTQM" agent="AGENT1"/>
            <destinationAgent QMgr="AGENTQM" agent="AGENT2"/>
            <transferSet priority="0">
              <metaDataSet>
                <metaData key="FileName">{FileName}</metaData>
              </metaDataSet>
              <item checksumMethod="MD5" mode="text">
                <source disposition="delete" recursive="false">
```

```

        <file>${FilePath}</file>
        </source>
        <destination type="queue">
          <queue persistent="true"
setMqProps="true">TEST.QUEUE@AGENTQM</queue>
        </destination>
      </item>
    </transferSet>
  </job>
  <name>Metadata_example</name>
</job>
</managedTransfer>
</request>
</transfer>
</task>
</tasks>
<originator>
  <hostName>mqjason.raleigh.ibm.com.</hostName>
  <userID>administrator</userID>
</originator>
</monitor:monitor>

```

Related tasks

[“Example: Setting IBM MQ message properties on a file-to-message transfer” on page 193](#)

You can use the **-qmp** parameter on the **fteCreateTransfer** command to specify whether IBM MQ message properties are set on the first message written to the destination queue by the transfer. IBM MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing IBM MQ Message Descriptor (MQMD) or MQRFH2 headers.

Related reference

[“fte:filecopy Ant task” on page 826](#)

The **fte:filecopy** task copies files between Managed File Transfer agents. The file is not deleted from the source agent.

[“fte:filemove Ant task” on page 830](#)

The **fte:filemove** task moves files between Managed File Transfer agents. When a file has been successfully transferred from the source agent to the destination agent, the file is deleted from the source agent.

Failure of a file-to-message transfer

If a file-to- message transfer fails after the agent has started writing file data to the destination queue, the agent writes a message to the queue to indicate to an application consuming the messages that a failure has occurred.

The message written to the destination queue if a failure occurs:

- Is blank
- Has the same IBM MQ group ID as the previous message written to the destination queue by the agent
- Has the IBM MQ LAST_MSG_IN_GROUP flag set
- Contains additional IBM MQ message properties, if message properties are enabled. For more information, see the topic [“Failure properties” on page 652](#).

Example

A transfer is requested by running the following command:

```

fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq RECEIVING_QUEUE
-qmp true -qs 1K /tmp/source1.txt

```

The file `source1.txt` is 48 KB. The transfer splits this file into 1 KB messages and writes these messages to the destination queue `RECEIVING_QUEUE`.

While the transfer is in progress, after the agent has written 16 messages to `RECEIVING_QUEUE`, a failure occurs at the source agent.

The agent writes a blank message to RECEIVING_QUEUE. In addition to the standard set of message properties, the blank message has the following message properties set:

```
usr.WMQFTEResultCode = 40
usr.WMQFTESupplement = BFGTR0036I: The transfer failed to complete successfully.
```

Transferring data from messages to files

The message-to-file feature of Managed File Transfer enables you to transfer data from one or more messages on an IBM MQ queue to a file, a data set (on z/OS), or a user file space. If you have an application that creates or processes IBM MQ messages, you can use the message-to-file capability of Managed File Transfer to transfer these messages to a file on any system in your Managed File Transfer network.

For information about file-to-message transfers, see [“Transferring data from files to messages” on page 189](#).



Attention: The source agent for a message-to-file transfer cannot be a protocol bridge agent or a Connect:Direct bridge agent.

You can transfer IBM MQ message data to a file. The following types of message-to-file transfer are supported:

- From a single message to a single file
- From multiple messages to a single file
- From multiple messages with the same IBM MQ group ID to a single file.
- From multiple messages to a single file, including a text or binary delimiter between the data from each message written to the file.

If you are transferring files from large messages, or many small messages, you might need to change some IBM MQ or Managed File Transfer properties. For more information about, see [“Guidance for setting MQ attributes and MFT properties associated with message size” on page 306](#).

V 9.0.1

From IBM MQ 9.0.1, in a message to file transfer, the source agent browses the messages from the source queue, unlike the destructive GET, in IBM MQ versions before IBM MQ 9.0.1. The messages are removed from the source queue after all messages (in a group if message grouping is used) have been browsed and data written to the destination file. This allows messages to remain in the source queue if a transfer fails or is canceled. Due to this change, authority to BROWSE must also be provided along with GET authority to run message to file transfers.

V 9.0.0.2

From IBM MQ 9.0.0 Fix Pack 2, Managed File Transfer is updated to restore the comparison check, previously removed by [APAR IT18213](#), of the transfer identifier and the value of the groupId attribute within the transfer request XML payload. If these two identifiers are equivalent, the source agent uses the identifier as a message identifier match option (as opposed to a group identifier match option) for the first MQGET attempt that is made on the input queue for the message-to-file transfer.

Configuring an agent to perform message-to-file transfers

By default agents cannot perform message to file, or file to message, transfers. To enable this function you must set the agent property enableQueueInputOutput to true.

About this task

If you attempt to perform a message to file transfer from a source agent that does not have the enableQueueInputOutput property set to true, the transfer fails. The transfer log message that is published to the coordination queue manager contains the following message:

```
BFGI00197E: An attempt to read from a queue was rejected by the source agent.
```


be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called AGENT_VENUS, which connects to QM_VENUS. The queue that AGENT_VENUS monitors is called START_QUEUE and is located on QM_VENUS. The agent polls the queue every 30 minutes.

When a complete group of messages is written to the queue the monitor task sends the group of messages to a file at one of a number of destination agents, all of which connect to the queue manager QM_MARS. The name of the file that the group of messages is transferred to is defined by the IBM MQ message property `usr.fileName` on the first message in the group. The name of the agent that the group of messages is sent to is defined by the IBM MQ message property `usr.toAgent` on the first message in the group. If the `usr.toAgent` header is not set, the default value to be used for the destination agent is AGENT_MAGENTA.

When you specify `useGroups="true"`, if you do not also specify `groupId="{GROUPLD}"`, the transfer just takes the first message on the queue. For example, if you are using variable substitution to generate the `fileName`, it is therefore possible that the contents of a `.txt` will not be correct. This is because the `fileName` is generated by the monitor, but the transfer actually gets a message that is not the one that should generate the file called `fileName`.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS"/>
    <destinationAgent agent="{toAgent}" QMgr="QM_MARS"/>
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <queue useGroups="true" groupId="{GROUPID}">START_QUEUE</queue>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/{fileName}.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The variables that are replaced with the values of IBM MQ message headers are highlighted in **bold**. This task XML is saved to the file `/home/USER1/task.xml`

2. Create a resource monitor to monitor the queue START_QUEUE.

Submit the following command:

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. A user or program writes a group of messages to the queue START_QUEUE.

The first message in this group has the following IBM MQ message properties set:

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. The monitor is triggered when the complete group is written. The agent substitutes the IBM MQ message properties into the task XML.

This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS"/>
    <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS"/>
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/larmer.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Results

The transfer that is defined by the task XML is performed. The complete group of messages that are read from the START_QUEUE by AGENT_VENUS is written to a file called /reports/larmer.rpt on the system where AGENT_VIOLET is running.

What to do next

Transferring each message to a separate file

If you want to monitor a queue and have every message transferred to a separate file, you can use a similar technique to the one described previously in this topic.

1. Create the monitor as described previously, specifying the **-tr completeGroups** parameter on the **fteCreateMonitor** command.
2. In the task XML specify the following:

```
<queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
```

However, when you put the messages onto the source queue, do not put them in an IBM MQ group. Add IBM MQ message properties to each message. For example, specify the `usr.fileName` property with a unique file name value for each message. This effectively causes the Managed File Transfer Agent to treat each message on the source queue as a separate group.

Example: Failing a message-to-file transfer using IBM MQ message properties

You can cause a message to file transfer to fail by setting the `usr.UserReturnCode` IBM MQ message property to a non-zero value. You can also specify supplementary information about the reason for the failure by setting the `usr.UserSupplement` IBM MQ message property.

About this task

In this example, a transfer is in progress between the queue `INPUT_QUEUE` and the file `/home/user/output.file`.

A user is creating messages and placing them on the queue `INPUT_QUEUE`. The source agent is consuming messages from the queue `INPUT_QUEUE` and is sending the transfer data to the destination agent. The destination agent is writing this data to the file `/home/user/output.file`.

The user writing messages to the queue `INPUT_QUEUE` wants to stop the transfer that is in progress and delete any data that has already been written to the destination file.

Procedure

1. The user writes a message to the queue `INPUT_QUEUE` that has the following IBM MQ message properties set:

```
usr.UserReturnCode=1
usr.UserSupplement="Cancelling transfer - sent wrong data."
```

2. The source agent reads the IBM MQ message properties and stops processing messages from the queue. The destination agent deletes any file data that has been written to the destination directory.
3. The source agent sends a transfer log message to the coordination queue manager reporting the transfer failure.

The message contains the following information:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d3120202020202020202020202020207e970d4920008702" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T21:28:09.593Z">progress</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1 </mqmdUserID>
  </originator>
  <transferSet index="0" size="1"
    startTime="2008-11-02T21:28:09.281Z"
    total="1">
    <item mode="binary">
      <source>
        <queue>INPUT_QUEUE@QM1</queue>
      </source>
      <destination exist="error">
        <file>/home/user/output.file</file>
      </destination>
      <status resultCode="1">
        <supplement>Cancelling transfer - sent wrong data.</supplement>
      </status>
    </item>
  </transferSet>
</transaction>
```

Listing MFT agents

You can list the Managed File Transfer agents registered with a particular queue manager using the command line or the IBM MQ Explorer.

About this task

To list agents using the command line, see [fteListAgents command](#).

To list agents using the IBM MQ Explorer, in the Navigator view click **Agents** under the coordination queue manager name.

If an agent is not listed by the **fteListAgents** command or is not displayed in the IBM MQ Explorer, use the diagnosis flowchart in the following topic to locate and fix the problem: [If your agent is not listed by the fteListAgents command](#).

Stopping an MFT agent

You can stop a Managed File Transfer agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

Before you begin

If you want to check the names of the agents associated with a queue manager, you can list agents by using the IBM MQ Explorer or the command line, see [fteListAgents command](#).

About this task

To stop an agent from the command line, see [fteStopAgent](#).

If you have configured your agent to run as a Windows service, running the **fteStopAgent** command also stops the Windows service. Alternatively, you can stop the agent by stopping the service by using the Windows Services tool. For more information, see the topic [“Starting an MFT agent as a Windows service” on page 147](#).

Stopping an MFT agent on z/OS

If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

A started task is used because it runs under a specific user ID and is not affected by users logging off.

Note:   Started tasks are typically run under an administrative user that might not have log-on privileges and so it is not possible to log on to the z/OS system as the user that the agent is running under. The **fteStartAgent**, **fteStopAgent**, **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, cannot be issued for that agent.

From IBM MQ 9.0.2 and IBM MQ 9.0.0 Fix Pack 1, the agent property **adminGroup** is available for use with Managed File Transfer agents on z/OS. You can define a security manager group, for example MFTADMIN and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the **adminGroup** property to be the name of this security manager group.

```
adminGroup=MFTADMIN
```

Members of this group can then issue the **fteStartAgent**, **fteStopAgent**, and **fteSetAgentTraceLevel** commands, and the **fteShowAgentDetails** command with the **-d** parameter specified, for the agent that is running as a started task.

For more information, see the **adminGroup** property in [“The MFT agent.properties file” on page 526](#).

Controlled agent shutdown by using the z/OS MODIFY command (F)

The **MODIFY** command allows you to stop an agent in a controlled way as an alternative to the **fteStopAgent** command. The agent completes any transfers currently in progress but the agent does not start any new transfers.

For example:

```
F job_name,APPL=STOP
```

where *job_name* is the job that the agent process is running under.

Immediate agent shutdown by using the z/OS STOP command (P)

The **STOP** command is equivalent to an immediate stop by using the **fteStopAgent** command with the **-i** parameter. The agent is stopped immediately even if the agent is currently transferring a file.

For example:

```
P job_name
```

where *job_name* is the job that the agent process is running under.

Related reference

[“Starting an MFT agent on z/OS” on page 150](#)

On z/OS, in addition to running the **fteStartAgent** command from a UNIX System Services session, you can start an agent as a started task from JCL without the need for an interactive session.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

The protocol bridge

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

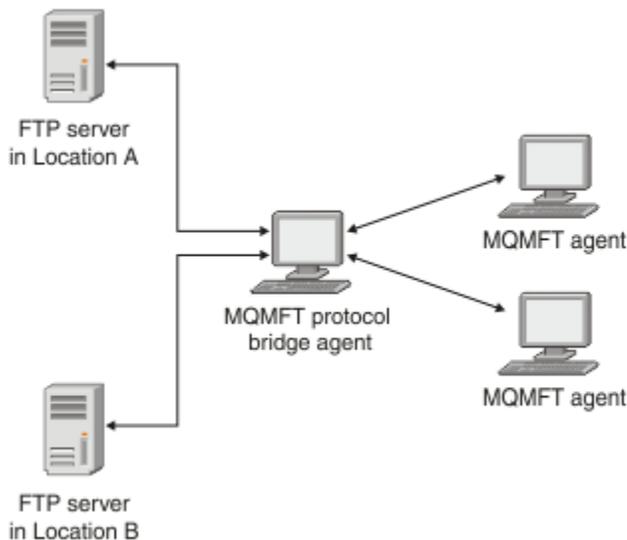
The protocol bridge is available as part of the Service component of Managed File Transfer. You can have multiple dedicated agents on a single system running MFT that connect to different file servers.

You can use a protocol bridge agent to transfer files to multiple endpoints simultaneously. MFT provides a file called `ProtocolBridgeProperties.xml` that you can edit to define the different protocol file servers that you want to transfer files to. The **fteCreateBridgeAgent** command adds the details of the default protocol file server to `ProtocolBridgeProperties.xml` for you. This file is described in [“Protocol bridge properties file format” on page 553](#).

You can use the protocol bridge agent to perform the following actions:

- Upload files from the MFT network to a remote server using FTP, FTPS, or SFTP.
- Download files from a remote server, using FTP, FTPS, or SFTP, to the MFT network

Note: The protocol bridge agent can support only FTP, FTPS, or SFTP servers that allow files to be accessed by their absolute file path. If a relative file path is specified in a transfer request, the protocol bridge agent will attempt to convert the relative path into an absolute file path based on the home directory used to login to the protocol server. Those protocol servers that allow access to files based only on the current directory are not supported by the protocol bridge agent.



The diagram shows two FTP servers, at different locations. The FTP servers are being used to exchange files with the Managed File Transfer agents. The protocol bridge agent is between the FTP servers and the rest of the MFT network, and is configured to communicate with both FTP servers.

Ensure that you have another agent in your MFT network in addition to the protocol bridge agent. The protocol bridge agent is a bridge to the FTP, FTPS, or SFTP server only and does not write transferred files to the local disk. If you want to transfer files to or from the FTP, FTPS, or SFTP server you must use the protocol bridge agent as the destination or source for the file transfer (representing the FTP, FTPS, or SFTP server) and another standard agent as the corresponding source or destination.

When you transfer files using the protocol bridge, the bridge must have permission to read the source or destination directory containing the files you want to transfer. For example, if you want to transfer files from the directory `/home/fte/bridge` that has execute permissions (`d--x--x--x`) only, any transfers you attempt from this directory fail with the following error message:

```
BFGBR0032E: Attempt to read filename from the protocol file server
has failed with server error 550. Failed to open file.
```

Configuring a protocol bridge agent

A protocol bridge agent is like a standard MFT agent. Create a protocol bridge agent by using the `fteCreateBridgeAgent` command. You can configure a protocol bridge agent using the `ProtocolBridgeProperties.xml` file, which is described in [“Protocol bridge properties file format”](#) on page 553. If you are using an earlier version, configure the agent using the specific protocol bridge properties described in [Advanced agent properties](#). For all versions, you can also configure a credential mapping as described in [“Mapping credentials for a file server”](#) on page 212. After you have configured a protocol bridge agent for a particular protocol file server, you can then use that agent for that purpose only.

Protocol bridge recovery

If the protocol bridge agent is unable to connect to the file server because the file server is unavailable, all file transfer requests are queued until the file server becomes available. If the protocol bridge agent is unable to connect to the file server because the agent is using the wrong credentials, the transfer fails and the transfer log message reflects this error. If the protocol bridge agent is ended for any reason, all requested file transfers are retained and continue when the protocol bridge is restarted.

During file transfer, files are typically written as temporary files at the destination and are then renamed when the transfer is complete. However, if the transfer destination is a protocol file server that is configured as limited write (users can upload files to the protocol file server but cannot change those uploaded files in any way; effectively users can write once only), transferred files are written to the

destination directly. This means that if a problem occurs during the transfer, the partially written files remain on the destination protocol file server and Managed File Transfer cannot delete or edit these files. In this situation, the transfer fails.

Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file

Define the properties of one or more protocol file servers that you want to transfer files to and from using the ProtocolBridgeProperties.xml file, which is provided by Managed File Transfer in the agent configuration directory.

About this task

The **fteCreateBridgeAgent** command creates the ProtocolBridgeProperties.xml file in the agent configuration directory *MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name*. The command also creates an entry in the file for the default protocol file server, if a default was specified when the command was run.

The message BFGCL0392I gives the location of the ProtocolBridgeProperties.xml file.

```
<?xml version="1.0" encoding="IBM-1047"?>
<!--
This ProtocolBridgeProperties.xml file determines the protocol servers that will be accessed by
the
MQMFT protocol bridge agent.

Each protocol server is defined using either a <tns:ftpServer>, <tns:ftpsServer>, or
<tns:sftpServer>
element - depending on the protocol used to communicate with the server. When the protocol
bridge agent participates in a managed file transfer it will determine which server to used
based on
the prefix (if any) present on the file path. For example a file path of 'server1:/home/user/
file.txt' would
be interpreted as a request to transfer /home/user/file.txt using 'server1'. The server name
is compared
to the 'name' attribute of each <tns:ftpServer>, <tns:ftpsServer> or <tns:sftpServer> element
in this
XML document and the first match is used to determine which protocol server the protocol bridge
agent will connect to. If no match is found then the managed file transfer operation will fail.

If a file path is not prefixed with a server name, for example '/home/user/file.txt' then this
XML
document can specify a default server to use for the managed file transfer. To specify a
default server use the <tns:defaultServer> element as the first element inside the
<tns:serverProperties>
element. The default server will be used whenever the protocol bridge agent participates in
a managed file transfer for file names which do not specify a prefix.

An optional <tns:limits> element can be specified within each server definition. This element
contains
attributes that govern the amount of resources used by each defined server.

An optional <tns:credentialsFile> element can be specified within each serverProperties
definition. This
element contains a path to a file containing credentials to be used when connecting to defined
servers.

An example ProtocolBridgeProperties.xml file is as follows:

<?xml version="1.0" encoding="UTF-8"?>
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
  ProtocolBridgeProperties.xsd">

  <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml"/>

  <tns:defaultServer name="myFTPserver"/>

  <tns:ftpServer name="myFTPserver" host="windows.hursley.ibm.com" port="1234"
  platform="windows"
  timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
  listFormat="unix" limitedWrite="false">
```

```

        <tns:limits maxListFileNames="100" maxListDirectoryLevels="999999999"
            maxReconnectRetry="2" reconnectWaitPeriod="10"
            maxSessions="60" socketTimeout="30"/>
    </tns:ftpServer>

    <tns:ftpsServer name="myFTPSserver" host="unix.hursley.ibm.com" platform="unix"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF8"
        listFormat="unix" limitedWrite="false" ftpsType="explicit"
        trustStore="C:\FTE\keystores\myFTPSserver\FTPSKeyStore.jks"
        trustStorePassword="password">

        <tns:limits maxReconnectRetry="10" connectionTimeout="10"/>

    </tns:ftpsServer>

    <tns:sftpServer name="mySFTPserver" host="windows.hursley.ibm.com" platform="windows"
        timeZone="Europe/London" locale="en_GB" fileEncoding="UTF-8"
        limitedWrite="false">

        <tns:limits connectionTimeout="60"/>

    </tns:sftpServer>
</tns:serverProperties>

```

This example shows the outermost <tns:serverProperties> element which must exist for the document to be valid, an optional <tns:defaultServer> element, as well as definitions for an FTP, FTPS and SFTP server.

The attributes of the <tns:ftpServer>, <tns:ftpsServer> and <tns:sftpServer> elements determine the characteristics of the connection established to the server. These attributes correspond to the command line parameters for the 'fteCreateBridgeAgent' command.

The following attributes are valid for all of the <tns:ftpServer>, <tns:ftpsServer> and <tns:sftpServer> elements: name, host, port, platform, fileEncoding, limitedWrite and controlEncoding.

The following attributes are valid for the <tns:ftpServer> and <tns:ftpsServer> elements: timeZone, locale, listFormat, listFileRecentDateFormat, listFileOldDateFormat, and monthShortNames.

The following attributes are valid for the <tns:ftpServer> element only: passiveMode

The following attributes are valid for the <tns:ftpsServer> element only: ftpsType, trustStore, trustStorePassword, trustStoreType, keyStore, keyStorePassword, keyStoreType, ccc, protFirst, auth, and connectTimeout.

The following attributes are valid for the <tns:limits> element within all of the <tns:ftpServer>, <tns:ftpsServer> and <tns:sftpServer> elements: maxListFileNames, maxListDirectoryLevels, maxReconnectRetry, reconnectWaitPeriod, maxSessions and socketTimeout

```

-->
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
ProtocolBridgeProperties.xsd">

```

```

    <!-- By default the location of the credentials file is in the home directory of the user
    that started the -->

```

```

    <!-- protocol bridge agent. If you wish to specify a different location use the
    credentialsFile element to -->

```

```

    <!-- do this. For

```

```

    example:

```

```

    <!-- <tns:credentialsFile path="/test/
    ProtocolBridgeCredentials.xml"/>

```

```

-->

```

```

        <tns:defaultServer name="WINMVSCA.HURSLEY.IBM.COM"/>
        <tns:ftpServer name="WINMVSCA.HURSLEY.IBM.COM" host="WINMVSCA.HURSLEY.IBM.COM"
platform="UNIX"
            timeZone="Europe/London" locale="en-GB" fileEncoding="US-ASCII"
            listFormat="unix" limitedWrite="false" />

```

```

    <!-- Define servers here -->
</tns:serverProperties>

```

The command can produce the following message: BFGCL0532I:

For this agent to function an additional credentials file must be created manually. By default this file is called ProtocolBridgeCredentials.xml and is located in the home directory of the user who starts the agent. For example, if this user started the agent the location would be: \$HOME/ProtocolBridgeCredentials.xml

If you use a credentials file:

1. See the following text for further information on how to create one.
2. The credentials file must be in a directory with restricted permissions. For example, there must be no read access for other users.
3. Specify the location of the directory for the credentials file in the \$HOME environment variable for the user ID of the started agent, or edit the ProtocolBridgeProperties.xml file and specify the location in:

```
<tns:credentialsFile path="/test/ProtocolBridgeCredentials.xml"/>
```

If you want to add further non-default protocol servers, edit this file to define their properties. This example adds an additional FTP server.

Note: The protocol bridge agent does not support file locking. This is because Managed File Transfer does not support the file locking mechanism on a file server.

Procedure

1. Define a protocol file server by inserting the following lines into the file as a child element of <tns:serverProperties>:

```
<tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234"
  platform="windows"
    timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
    listFormat="unix" limitedWrite="false" >
<tns:limits maxListFileNames="10" maxListDirectoryLevels="500"/>
```

2. Then change the value of the attributes:

- name is the name of your protocol file server
- host is the host name or IP address of the protocol file server
- port is the port number of the protocol file server
- platform is the platform that the protocol file server runs on
- timeZone is the time zone that the protocol file server runs in
- locale is the language used on the protocol file server
- fileEncoding is the character encoding of the protocol file server
- listFormat is the file listing format returned from the protocol file server
- limitedWrite determines whether to follow the default mode when writing to a file server, which is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The limitedWrite attribute and the doNotUseTempOutputFile agent property are used together in the case of protocol bridge agents. If you want to use temporary files, then you must not set the value of doNotUseTempOutputFile, and you must set the value of limitedWrite to false. Any other combination of settings means that temporary files will not be used.
- maxListFileNames is the maximum number of names collected when scanning a directory on the protocol file server for file names.
- maxListDirectoryLevels is the maximum number of directory levels to recurse when scanning a directory on the protocol file server for file names.

For more details about these attributes, including whether they are required or optional and their default values, see [“Protocol bridge properties file format” on page 553](#).

Related reference

[“Protocol bridge properties file format” on page 553](#)

The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

[“Regular expressions used by MFT” on page 633](#)

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

ProtocolBridgePropertiesExit2: Looking up protocol file server properties

If you have a large number of protocol file servers, you can implement the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2` interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in preference to maintaining a `ProtocolBridgeProperties.xml` file. Managed File Transfer provides a sample user exit that looks up protocol file server properties.

Configuring user exits that look up protocol bridge properties

About this task

Any user exit that looks up protocol bridge properties must implement the interface `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2`. For more information, see [“ProtocolBridgePropertiesExit2.java interface” on page 880](#).

You can chain multiple protocol server properties exits together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgePropertiesExitClasses` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the `getProtocolServerProperties` methods of all of the exits. If the method returns a properties object as the result code, this value is the returned result and the `getProtocolServerProperties` methods of the subsequent exits are not called. If the method returns a value of null as the result code, the `getProtocolServerProperties` method of the next exit is called. If there is no subsequent exit, the null result is returned. An overall result code of null is considered as a lookup failure by the protocol bridge agent.

You are recommended to use the `ProtocolBridgePropertiesExit2.java` interface, but for information about the `ProtocolBridgePropertiesExit.java` interface, see [“ProtocolBridgePropertiesExit: Looking up protocol file server properties” on page 211](#).

To run your exit, complete the following steps:

Procedure

1. Compile the protocol server properties user exit.
2. Create a Java archive (JAR) file containing the compiled exit and its package structure.
3. Put the JAR file containing the exit class in the `exits` directory of the protocol bridge agent. This directory is found in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name` directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgePropertiesExitClasses`. For the value of this property, specify a comma-separated list of classes that implement a protocol bridge server properties user exit. The exit classes are called in the order that they are specified in this list. For more information, see [“The MFT agent.properties file” on page 526](#).

5. You can optionally specify the `protocolBridgePropertiesConfiguration` property. The value you specify for this property is passed in as a `String` to the `initialize()` method of the exit classes specified by `protocolBridgePropertiesExitClasses`. For more information, see [“The MFT agent.properties file”](#) on page 526.

Using the sample user exit

About this task

A sample user exit that looks up protocol bridge properties is provided in the `MQ_INSTALLATION_PATH/mqft/samples/protocolBridge` directory and in the topic [“Sample protocol bridge properties user exit”](#) on page 255.

The `SamplePropertiesExit2.java` exit reads a properties file that contains properties for protocol servers. The format of each entry in the properties file is as follows:

```
serverName=type://host:port
```

The location of the properties file is taken from the protocol bridge agent property `protocolBridgePropertiesConfiguration`.

To run the sample user exit, complete the following steps:

Procedure

1. Compile the `SamplePropertiesExit2.java` file.
2. Create a JAR file containing the compiled exit and its package structure.
3. Put the JAR file in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent/exits` directory.
4. Edit the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name/agent.properties` file to contain the line:

```
protocolBridgePropertiesExitClasses=SamplePropertiesExit2
```

5. Create a protocol bridge properties file, for example `protocol_bridge_properties.properties`, in the directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent`. Edit this file to include entries in the format:

```
serverName=type://host:port
```

6. Edit the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent/agent.properties` file to contain the line:

```
protocolBridgePropertiesConfiguration=MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent/protocol_bridge_properties.properties
```

You must use the absolute path to the `protocol_bridge_properties.properties` file.

7. Start the protocol bridge agent by using the **fteStartAgent** command.

Related concepts

[“The protocol bridge”](#) on page 204

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

Related reference

[“ProtocolBridgePropertiesExit2.java interface” on page 880](#)

[“Sample protocol bridge properties user exit” on page 255](#)

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

[“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)” on page 375](#)

The **`fteCreateBridgeAgent`** command creates a Managed File Transfer protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

ProtocolBridgePropertiesExit: Looking up protocol file server properties

If you have a large number of protocol file servers, you can implement the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit` interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in preference to maintaining a `ProtocolBridgeProperties.xml` file. You are recommended to use the `ProtocolBridgePropertiesExit2.java` interface but the `ProtocolBridgePropertiesExit.java` interface is also supported. If you have an existing implementation of the `ProtocolBridgePropertiesExit.java` interface from IBM WebSphere MQ File Transfer Edition, you can use it in IBM WebSphere MQ 7.5 or later. The new `getCredentialLocation` method in `ProtocolBridgePropertiesExit2.java` uses the default location of the `ProtocolBridgeCredentials.xml` file, which is your home directory.

Configuring user exits that look up protocol bridge properties

Any user exit that looks up protocol bridge properties must implement the interface `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit`. For more information, see [“ProtocolBridgePropertiesExit.java interface” on page 212](#).

You can chain multiple protocol server properties exits together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgePropertiesExitClasses` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the `getProtocolServerProperties` methods of all of the exits. If the method returns a properties object as the result code, this value is the returned result and the `getProtocolServerProperties` methods of the subsequent exits are not called. If the method returns a value of null as the result code, the `getProtocolServerProperties` method of the next exit is called. If there is no subsequent exit, the null result is returned. An overall result code of null is considered as a lookup failure by the protocol bridge agent.

To run your exit, complete the following steps:

1. Compile the protocol server properties user exit.
2. Create a Java archive (JAR) file containing the compiled exit and its package structure.
3. Put the JAR file containing the exit class in the `exits` directory of the protocol bridge agent. This directory is found in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name` directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgePropertiesExitClasses`. For the value of this property, specify a comma-separated list of classes that implement a protocol bridge server properties user exit. The exit classes are called in the order that they are specified in this list. For more information, see [“The MFT agent.properties file” on page 526](#).
5. You can optionally specify the `protocolBridgePropertiesConfiguration` property. The value you specify for this property is passed in as a String to the `initialize()` method of the exit classes

specified by `protocolBridgePropertiesExitClasses`. For more information, see [“The MFT agent.properties file” on page 526](#).

ProtocolBridgePropertiesExit.java interface

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *     The values of properties defined for the protocol bridge.
     *     These values can only be read, they cannot be updated by the
     *     implementation.
     * @return {@code true} if the initialization is successful and {@code
     *     false} if unsuccessful. If {@code false} is returned from an exit
     *     the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *     The name of the protocol server whose properties are to be
     *     returned. If a null or a blank value is specified, properties
     *     for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *     if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
        final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *     The values of properties defined for the protocol bridge.
     *     These values can only be read, they cannot be updated by the
     *     implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);
}
```

Mapping credentials for a file server

Map user credentials in Managed File Transfer to user credentials on the file server by using the default credential mapping function of the protocol bridge agent or by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping.

Mapping credentials for a file server by using the ProtocolBridgeCredentials.xml file

Map user credentials in Managed File Transfer to user credentials on the file server by using the default credential mapping function of the protocol bridge agent. Managed File Transfer provides an XML file that you can edit to include your credential information.

About this task

The ProtocolBridgeCredentials.xml file must be manually created by the user. By default, the location of this file is the home directory of the user who started the protocol bridge agent, but this can be stored anywhere on the file system accessible by the agent. To specify a different location, add the <credentialsFile> element to the ProtocolBridgeProperties.xml file. For example,

```
<tns:credentialsFile path="/example/path/to/ProtocolBridgeCredentials.xml"/>
```

Before you can use a protocol bridge agent, set up credential mapping by editing this file to include host, user, and credential information. For more information and samples, see [“Protocol bridge credentials file format”](#) on page 549.

z/OS When you create the ProtocolBridgeCredentials.xml file on a z/OS platform using IBM WebSphere MQ 7.5 or earlier, you must set a file tag before you edit the file. Run the following command to mark the file as having ASCII contents:

```
chtag -t -c IS08859-1 ProtocolBridgeCredentials.xml
```

Note: **z/OS** On z/OS, you can store the protocol bridge credential file on a data set, where the name of the .xml file can be specified by the user.

Procedure

1. Edit the line <tns:server name="server name"> to change the value of the name attribute to the server name in the ProtocolBridgeProperties.xml file.

Protocol bridge agents that are created for IBM WebSphere MQ File Transfer Edition 7.0.4 and earlier do not have a ProtocolBridgeProperties.xml file (or related user exits), so for IBM WebSphere MQ File Transfer Edition 7.0.4 Fix Pack 1 and later the server name is automatically assigned the server's host name. Therefore, if you use an updated ProtocolBridgeCredentials.xml file with <server> entries, a name corresponding to the server's host name will match.

You can use the pattern attribute to specify that you used a server name that contains wildcards or regular expressions. For example,

```
<tns:server name="serverA*" pattern="wildcard">
```

2. Insert user ID and credential information into the file as child elements of <tns:server>.

You can insert one or many of the following elements into the file:

- If the protocol file server is an FTP, FTPS, or SFTP server, you can use passwords to authenticate the user requesting the transfer. Insert the following lines into the file:

```
<tns:user name="FTE User ID"
  serverUserId="Server User ID"
  serverPassword="Server Password">
</tns:user>
```

Then change the value of the attributes.

- name is a Java regular expression to match the MQMD user ID associated with the MFT transfer request

- `serverUserId` is the value that is passed to the protocol file server as the login user ID. If the `serverUserId` attribute is not specified, the MQMD user ID associated with the MFT transfer request is used instead
- `serverPassword` is the password that is associated with the `serverUserId`.

The `name` attribute can contain a Java regular expression. The credential mapper attempts to match the MQMD user ID of the MFT transfer request to this regular expression. The protocol bridge agent attempts to match the MQMD user ID to the regular expression in the `name` attribute of the `<tns:user>` elements in the order that the elements exist in the file. When a match is found the protocol bridge agent does not look for more matches. If a match is found, the corresponding `serverUserId` and `serverPassword` values are passed to the protocol file server as the login user ID and password. The MQMD user ID matches are case-sensitive.

- If the protocol file server is an SFTP server, you can use public and private keys to authenticate the user requesting the transfer. Insert the following lines into the file and change the value of the attributes. The `<tns:user>` element can contain one or many `<tns:privateKey>` elements.

```
<tns:user name="FTE User ID"
  serverUserId="Server User ID"
  hostKey="Host Key">
  <tns:privateKey associationName="association"
    keyPassword="Private key password">
    Private key file text
  </tns:privateKey>
</tns:user>
```

- `name` is a Java regular expression to match the MQMD user ID associated with the MFT transfer request
- `serverUserId` is the value that is passed to the protocol file server as the login user ID. If the `serverUserId` attribute is not specified, the MQMD user ID associated with the MFT transfer request is used instead
- `hostKey` is the expected key that is returned from the server when logging on
- `key` is the private key of the `serverUserId`
- `keyPassword` is the password for the key to generate public keys
- `associationName` is a value that is used to identify for trace and logging purposes

The `name` attribute can contain a Java regular expression. The credential mapper attempts to match the MQMD user ID of the MFT transfer request to this regular expression. The protocol bridge agent attempts to match the MQMD user ID to the regular expression in the `name` attribute of the `<tns:user>` elements in the order that the elements exist in the file. When a match is found the protocol bridge agent does not look for more matches. If a match is found, the corresponding `serverUserId` and `key` values are used to authenticate the MFT user with the protocol file server. The MQMD user ID matches are case-sensitive.

For more information about using private keys with a protocol bridge agent, see [“Example: How to configure a protocol bridge agent to use private key credentials with a UNIX SFTP server”](#) on page 217.

Note:  z/OS

When the transfer request is written to the command queue, the MQMD user ID might be converted to uppercase if the source agent command queue is on a z/OS or IBM i system. As a result the MQMD user ID for the same originating user might arrive at the credentials exit in the original case or converted to uppercase depending on the source agent that is specified in the transfer request. The default credential mapping exit performs case-sensitive matches against the supplied MQMD user ID, which you might need to allow for in the mapping file.

Mapping credentials for a file server by using exit classes

If you do not want to use the default credential mapping function of the protocol bridge agent, you can map user credentials in Managed File Transfer to user credentials on the file server by writing your own

user exit. Managed File Transfer provides a sample user exit that performs user credential mapping. If you configure credential mapping user exits, they take the place of the default credential mapping function.

Configuring protocol bridge credential user exits

About this task

A user exit for mapping protocol bridge credentials must implement one of the following interfaces:

- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit`, which allows a protocol bridge agent to transfer files to and from one default protocol file server
- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2`, which allows you to transfer files to and from multiple endpoints.

The `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2` interface contains the same function as `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit` and also includes extended function. For more information, see [“ProtocolBridgeCredentialExit.java interface” on page 878](#) and [“ProtocolBridgeCredentialExit2.java interface” on page 880](#).

The credential exits can be chained together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgeCredentialConfiguration` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the `mapMQUserId` methods of all of the exits as follows:

- If the method returns a value of `USER_SUCCESSFULLY_MAPPED` or `USER_DENIED_ACCESS` as the result code, this value is the returned result and the `mapMQUserId` methods of the subsequent exits are not called.
- If the method returns a value of `NO_MAPPING_FOUND` as the result code, the `mqMQUserId` method of the next exit is called.
- If there is no subsequent exit, the `NO_MAPPING_FOUND` result is returned.
- An overall result code of `USER_DENIED_ACCESS` or `NO_MAPPING_FOUND` is considered as a transfer failure by the bridge agent.

To run your exit, complete the following steps:

Procedure

1. Compile the protocol bridge credential user exit.
2. Create a Java archive (JAR) file that contains the compiled exit and its package structure.
3. Place the JAR file that contains the exit class in the `exits` directory of the bridge agent. The directory is in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name` directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgeCredentialExitClasses`. For the value of this property, specify a comma-separated list of classes that implement a protocol bridge credential exit routine. The exit classes are called in the order that they are specified in this list. For more information, see [“The MFT agent.properties file” on page 526](#).
5. Edit the property file of the protocol bridge agent to include:

```
exitClassPath=IBM MQ
installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
ame\exits\SampleCredentialExit.jar
```

The `agent.properties` file for an agent is in your `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/bridge_agent_name` directory.

If you change the `agent.properties` file, you must restart the agent to pick up the changes.

6. You can optionally specify the `protocolBridgeCredentialConfiguration` property. The value that you specify for this property is passed in as a `String` object to the `initialize()` method of the exit classes specified by `protocolBridgeCredentialExitClasses`. For more information, see [“The MFT agent.properties file”](#) on page 526.
7. Start the protocol bridge agent with the **`fteStartAgent`** command.

Using the sample user exit

About this task

A sample protocol bridge credential exit is provided in the `MQ_INSTALLATION_PATH/mqft/samples/protocolBridge` directory and in the topic [“Sample protocol bridge credential user exit”](#) on page 253. This sample is based on the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit` interface.

The `SampleCredentialExit.java` exit reads a properties file that maps the MQMD user IDs associated with transfer requests to server user IDs and server passwords. The location of the properties file is taken from the protocol bridge agent property `protocolBridgeCredentialConfiguration`.

To run the sample user exit, complete the following steps:

Procedure

1. Compile the `SampleCredentialExit.java` file.
2. Create a JAR file that contains the compiled exit and its package structure.
3. Place the JAR file in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name/exits` directory.
4. Edit the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name/agent.properties` file to contain the line:

```
protocolBridgeCredentialExitClasses=SampleCredentialExit
```

5. Edit the property file of the protocol bridge agent to include:

```
exitClassPath=IBM MQ
installation_directory\mqft\config\configuration_queue_manager\agents\protocol_bridge_agent_n
ame\exits\SampleCredentialExit.jar
```

The `agent.properties` file for an agent is in your `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

If you change the `agent.properties` file, you must restart the agent to pick up the changes.

6. Create a credential properties file (`credentials.properties`) in the directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent` and edit it to include entries in the format:

```
mqUserId=serverUserId,serverPassword
```

7. Edit the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name/agent.properties` file to contain the line:

```
protocolBridgeCredentialConfiguration=MQ_DATA_PATH/mqft/
config/coordination_queue_manager/agents/bridge_agent_name/credentials.properties
```

You must use the absolute path to the `credentials.properties` file.

8. Start the protocol bridge agent by using the **`fteStartAgent`** command.

Example: How to configure a protocol bridge agent to use private key credentials with a UNIX SFTP server

This example demonstrates how you can generate and configure the `ProtocolBridgeCredentials.xml` file. This example is a typical example and the details might vary according to your platform, but the principles remain the same.

About this task

Procedure

1. Generate a public and private key to be used to authenticate with the SFTP server.

For example, on a Linux host system, you can use the tool **ssh-keygen**, supplied as part of the 'openssh' package, to create the public/private key pair.

By default, with no arguments, the **ssh-keygen** command prompts for a location and passphrase for the two key files, which defaults to the names:

```
id_rsa      <-- Private key
id_rsa.pub  <-- Public key
```



Attention: If you are using the **ssh-keygen** command from a recent version of OpenSSH, such as that supplied with RHEL 8, the key format used is not compatible with the protocol bridge agent, and transfer attempts to the SFTP server fail with the message:

```
BFGBR0216E: Authentication to protocol server 'sftp.host.address' failed
because of invalid private key.
```

To create a compatible private key with these newer versions of OpenSSH, specify the key format with the following argument to the **ssh-keygen** command:

```
ssh-keygen -m PEM
```

The contents of the `id_rsa` private key then have the first and last lines of:

```
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
```

which is compatible with the protocol bridge agent.

2. Copy the entire contents of the `id_rsa.pub` file into the `~/.ssh/authorized_keys` file of the SFTP user on the SFTP server.

Ensure that the file permissions on this file and the `~/.ssh` directory are set appropriately for the SFTP server to allow key authentication. These permissions are typically:

```
~/.ssh          Mode 700
~/.ssh/authorized_keys  Mode 600
```

3. Managed File Transfer requires a host ssh fingerprint generated using the MD5 algorithm. Run one of the following commands to obtain the host ssh fingerprint of the SFTP server.

- For Red Hat® Enterprise Linux version 6.x and below, and Linux Ubuntu 14.04, run the following command:

```
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
```

- Starting with Red Hat Enterprise Linux version 7.x, Linux Ubuntu 16.04 and SuSE Linux 12.4, the `ssh-keygen` command generates, by default, the ssh fingerprint using the SHA56 algorithm. To generate the ssh fingerprint using the MD5 algorithm, run the following command:

```
ssh-keygen -l -E MD5 -f /etc/ssh/ssh_host_rsa_key.pub
```

The output of the command will be similar to the following example:

```
2048 MD5:64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27 no comment (RSA)
```

Extract the hexadecimal portion only of the output to use as the `hostKey` in the `ProtocolBridgeCredentials.xml` file (see step “4” on page 218). Therefore, in this example, you would extract `64:39:f5:49:41:10:55:d2:0b:81:42:5c:87:62:9d:27`.

- On the protocol bridge agent system, edit the `ProtocolBridgeCredentials.xml` file. Substitute the values shown in italics in the following example with your own values:

```
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
ProtocolBridgeCredentials.xsd ">

<tns:agent name="Agent_name">

<tns:server name="SFTP_name">

<tns:user name="mq_User_ID" serverUserId="SFTP_user_ID"
hostKey="ssh_host_finger">
<tns:privateKey associationName="name" keyPassword="pass_phrase">
Complete contents of the id_rsa file including the entries
-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----
</tns:privateKey>
</tns:user>

</tns:server>
</tns:agent>
</tns:credentials>
```

where:

- `Agent_name` is the name of the protocol bridge agent.
- `SFTP_host_name` is the name of the SFTP server as shown in the `ProtocolBridgeProperties.xml` file.
- `mq_User_ID` is the MQMD user ID associated with the transfer request.
- `SFTP_user_ID` is the SFTP user ID as used in step 2. It is the value passed to the SFTP serve as the login user ID.
- `ssh_host_finger` is the fingerprint collected in step 3.
- `name` is a name that you can specify to be used for trace and logging purposes.
- `pass_phrase` is the pass phrase you provided in the `ssh-keygen` in step 1.
- Complete contents of the id_rsa file* is the complete contents of the generated `id_rsa` file from step 1. To prevent a connection error, ensure that you include both of the following entries:

```
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----
```

You can add additional keys by duplicating the `<tns:privatekey>` element.

- Start the protocol bridge agent if the agent is not already started. Alternatively, the protocol bridge agent periodically polls the `ProtocolBridgeCredentials.xml` file and pick up the changes.

Configuring a protocol bridge for an FTPS server

Configure an FTPS server in a similar way as you configure an FTP server: create a bridge agent for the server, define the server properties, and map user credentials.

About this task

To configure an FTPS server, complete the following steps:

Procedure

1. Create a protocol bridge agent for the FTPS server by using the **fteCreateBridgeAgent** command. The parameters that are applicable to FTP are also applicable to FTPS but there are also three required parameters specific to FTPS:
 - a) The **-bt** parameter. Specify FTPS as the value of this parameter.
 - b) The **-bts** parameter for the truststore file. The command assumes that only server authentication is required and you must specify the location of the truststore file.

The explicit form of the FTPS protocol is configured by the **fteCreateBridgeAgent** command by default but you can configure the implicit form by changing the protocol bridge properties file. The protocol bridge always connects to FTPS servers in passive mode.

For more information about the **fteCreateBridgeAgent** command, see [“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)”](#) on page 375.

If you need instructions about how to create truststore files, see the IBM Developer article, [Configuring Secure Sockets Layer connectivity in IBM WebSphere MQ File Transfer Edition](#), or see the information about the keytool at the [Oracle keytool documentation](#).

2. Define the FTPS server properties within an `<ftpsServer>` element in the protocol bridge properties file: `ProtocolBridgeProperties.xml`. For more information, see [“Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file”](#) on page 206. You can also enable client authentication by editing the protocol bridge properties file. For details of all the configuration options, see [“Protocol bridge properties file format”](#) on page 553.
3. Map user credentials in Managed File Transfer to user credentials on the FTPS server either by using the default credential mapping function of the protocol bridge agent or by writing your own user exit. For more information, see [“Mapping credentials for a file server”](#) on page 212.
4. By default, the truststore file is configured as having the JKS format; if you want to change the format, edit the protocol bridge properties file.

Example

An example entry for an FTPS server in the protocol bridge properties file is shown as follows:

```
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
  ProtocolBridgeProperties.xsd">
  <tns:defaultServer name="ftpserver.mycompany.com"/>

  <tns:ftpsServer name="ftpserver.mycompany.com" host="ftpserver.mycompany.com" port="990"
  platform="windows"
  timeZone="Europe/London" locale="en_US" fileEncoding="UTF8"
  listFormat="unix" limitedWrite="false"
  trustStore="c:\mydirec\truststore.jks"/>

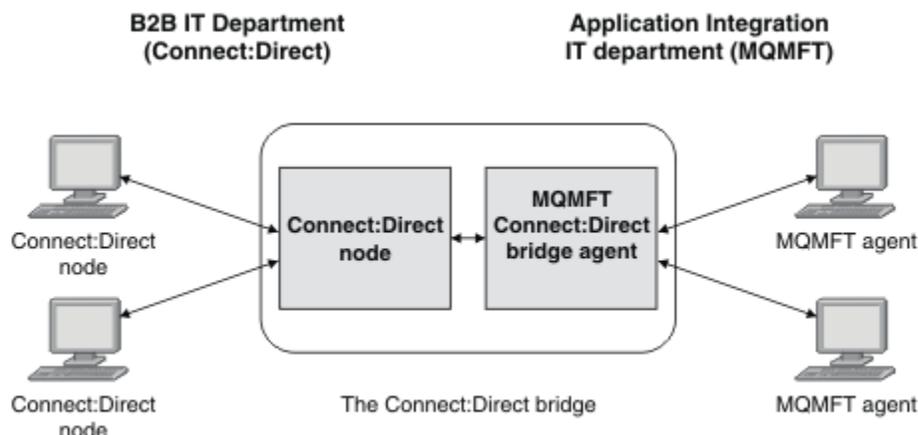
  <!-- Define servers here -->
</tns:serverProperties>
```

What to do next

For information about the parts of the FTPS protocol that are supported and, which are not supported, see [“FTPS server support by the protocol bridge”](#) on page 639.

The Connect:Direct bridge

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.



The diagram shows an MFT Connect:Direct bridge between two departments, the B2B IT department and the Application Integration IT department. The B2B IT department uses Connect:Direct to transfer files to and from the company's business partners. The Application Integration IT department uses IBM MQ as its messaging infrastructure and so has recently chosen Managed File Transfer as its file transfer solution.

Using the MFT Connect:Direct bridge, the two departments can transfer files between the Connect:Direct network in the B2B IT department and the MFT network in the Application Integration IT department. The Connect:Direct bridge is a component of Managed File Transfer, which includes an MFT agent that communicates with a Connect:Direct node. The MFT agent is dedicated to transfers with the Connect:Direct node, and is known as the Connect:Direct bridge agent.

The Connect:Direct bridge is available as part of the Service and Agent components of Managed File Transfer, and can be used for the following tasks:

1. Use Managed File Transfer commands to initiate a transfer of a file, or multiple files, from an MFT agent to a Connect:Direct node.
2. Use Managed File Transfer commands to initiate a transfer of a file, or multiple files, from a Connect:Direct node to an MFT agent.
3. Use Managed File Transfer commands to initiate a file transfer that starts a user-defined Connect:Direct process.
4. Use Connect:Direct process to submit an MFT file transfer request.

A Connect:Direct bridge can transfer files to or from only Connect:Direct nodes. The Connect:Direct bridge can transfer files to or from its local file system only as part of a transfer submitted by a Connect:Direct process.

z/OS You can use the Connect:Direct bridge to transfer to or from a data set that is located on a Connect:Direct node on a z/OS system. There are some differences in behavior compared to data set transfers that only involve Managed File Transfer agents. For more information, see [“Transferring data sets to and from Connect:Direct nodes”](#) on page 609.

Supported platforms

The Connect:Direct bridge is made up of an MFT Connect:Direct bridge agent and a Connect:Direct node. The agent is supported on Windows and Linux for System x. The node is supported on the platforms that are supported for IBM Sterling Connect:Direct for Windows and IBM Sterling Connect:Direct for UNIX. For instructions on creating the Connect:Direct bridge agent and configuring a Connect:Direct node for the agent to communicate with, see [“Configuring the Connect:Direct bridge” on page 135](#).

The Connect:Direct bridge can transfer files to and from Connect:Direct nodes that are running as part of a Connect:Direct for Windows or Connect:Direct for UNIX , or Connect:Direct for z/OS Service installation. For details of the versions of Connect:Direct that are supported, see the web page [IBM MQ System Requirements](#).

The agent and node that make up the Connect:Direct bridge must be on the same system, or have access to the same file system, for example through a shared NFS mount. This file system is used to temporarily store files during file transfers that involve the Connect:Direct bridge, in a directory defined by the **cdTmpDir** parameter. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to address this directory using the same path name. For example, if the agent and node are on separate Windows systems, the systems must use the same drive letter to mount the shared file system. The following configurations allow the agent and the node to use the same path name:

- The agent and node are on the same system, which is either running Windows or Linux for System x
- The agent is on Linux for System x, and the node is on UNIX
- The agent is on one Windows system, and the node is on another Windows system

The following configurations do not allow the agent and the node to use the same path name:

- The agent is on Linux for System x, and the node is on Windows
- The agent is on Windows, and the node is on UNIX

Consider this restriction when planning your installation of the Connect:Direct bridge.

Transferring a file to a Connect:Direct node

You can transfer a file from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge. Specify a Connect:Direct node as the destination of the transfer by specifying the Connect:Direct bridge agent as the destination agent and specifying the destination file in the form *connect_direct_node_name:file_path*.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. For more information, see [“Configuring the Connect:Direct bridge” on page 135](#).

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The source agent is called FTE_AGENT and can be any version of WMQFTE. The destination Connect:Direct node is called CD_NODE1. The file to be transferred is located at the file path `/home/helen/file.log` on the system where FTE_AGENT is located. The file is transferred to the file path `/files/data.log` on the system where CD_NODE1 is running.

Procedure

1. Use the `fteCreateTransfer` command with the value for the **-df** (destination file) parameter in the form *connect_direct_node_name:file_path* and the value of the **-da** (destination agent) parameter specified as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by `connect_direct_node_name` is the node that you want the file to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE  
-df CD_NODE1:/files/data.log /home/helen/file.log
```

For more information, see [“fteCreateTransfer: start a new file transfer”](#) on page 417.

2. The source agent FTE_AGENT transfers the file to the Connect:Direct bridge agent CD_BRIDGE. The file is temporarily stored on the system where the Connect:Direct bridge agent is running, in the location defined by the `cdTmpDir` agent property. The Connect:Direct bridge agent transfers the file to the Connect:Direct node CD_NODE1.

Related concepts

[“The Connect:Direct bridge”](#) on page 220

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related tasks

[“Transferring a file from a Connect:Direct node”](#) on page 222

You can transfer a file from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form `connect_direct_node_name:file_path`.

Related reference

[“The MFT agent.properties file”](#) on page 526

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring a file from a Connect:Direct node

You can transfer a file from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form `connect_direct_node_name:file_path`.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. See [“Configuring the Connect:Direct bridge”](#) on page 135.

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The destination agent is called FTE_AGENT and can be any version of Managed File Transfer. The source Connect:Direct node is called CD_NODE1. The file to be transferred is located at the file path `/home/brian/in.file` on the system where CD_NODE1 is located. The file is transferred to the file path `/files/out.file` on the system where FTE_AGENT is running.

Procedure

Use the **fteCreateTransfer** command with the value for the source specification in the form `connect_direct_node_name:file_path` and the value of the **-sa** parameter specified as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the file to be transferred from, not the Connect:Direct node that operates as part of the Connect:Direct bridge. For example:

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_AGENT  
-df /files/out.file CD_NODE1:/home/brian/in.file
```

For more information, see [“fteCreateTransfer: start a new file transfer”](#) on page 417.

Results

The Connect:Direct bridge agent CD_BRIDGE requests the file from the Connect:Direct node CD_NODE1. The Connect:Direct node sends the file to the Connect:Direct bridge. While the file is being transferred from the Connect:Direct node, the Connect:Direct bridge stores the file temporarily in the location defined by the cdTmpDir agent property. When the file has finished transferring from the Connect:Direct node to the Connect:Direct bridge, the Connect:Direct bridge then sends the file to the destination agent FTE_AGENT and deletes the file from the temporary location.

Related concepts

[“The Connect:Direct bridge”](#) on page 220

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related reference

[“The MFT agent.properties file”](#) on page 526

Each Managed File Transfer Agent has its own properties file, *agent.properties*, that must contain the information that an agent uses to connect to its queue manager. The *agent.properties* file can also contain properties that alter the behavior of the agent.

Transferring a data set to a Connect:Direct node on z/OS

You can transfer a data set from a Managed File Transfer agent on z/OS to a Connect:Direct node on z/OS by using a Connect:Direct bridge that is located on a Windows or Linux system.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. See [“Configuring the Connect:Direct bridge”](#) on page 135.

About this task

In this example, the parameter **-df** is used to specify the destination of the transfer. The parameter **-df** is valid for use when the source agent of the transfer is any version of Managed File Transfer. If the source agent is IBM WebSphere MQ File Transfer Edition 7.0.4 or later, you can use the **-ds** parameter instead. The source agent is called FTE_ZOS1 and is an IBM WebSphere MQ File Transfer Edition 7.0.3 agent. The Connect:Direct bridge agent is called CD_BRIDGE and is located on a Linux system. The destination Connect:Direct node is called CD_ZOS2. Both the source agent and the destination Connect:Direct node are located on z/OS systems. The data set to be transferred is located at //FTEUSER.SOURCE.LIB on the system where FTE_ZOS1 is located. The data set is transferred to the data set //CDUSER.DEST.LIB on the system where CD_ZOS2 is located.

Procedure

1. Use the fteCreateTransfer command with the value for the **-df** parameter in the form: *connect_direct_node_name:data_set_name;attributes* and the value of the **-da** (destination agent) parameter specified as the name of the Connect:Direct bridge agent.

The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the data set to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

The data set name specified by *data_set_name* must be absolute, not relative. Connect:Direct does not prefix the data set name with the name of the user.

```
fteCreateTransfer -sa FTE_ZOS1 -sm QM_ZOS
                 -da CD_BRIDGE -dm QM_BRIDGE
                 -df CD_ZOS2: //'CDUSER.DEST.LIB;BLKSIZE(8000);LRECL(80)'
                 //'FTEUSER.SOURCE.LIB'
```

For more information, see [“fteCreateTransfer: start a new file transfer” on page 417](#).

2. The source agent FTE_ZOS1 transfers the data in the data set to the Connect:Direct bridge agent CD_BRIDGE. The data is temporarily stored as a flat file on the system where the Connect:Direct bridge agent is running, in the location defined by the cdTmpDir agent property. The Connect:Direct bridge agent transfers the data to the Connect:Direct node CD_ZOS2. When the transfer is complete, the flat file is deleted from the system where the Connect:Direct bridge agent is running.

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related reference

[“Transferring data sets to and from Connect:Direct nodes” on page 609](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[“Mappings between Connect:Direct process statement parameters and BPXWDYN keys” on page 611](#)

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

Transferring multiple files to a Connect:Direct node

You can transfer multiple files from a Managed File Transfer Agent to a Connect:Direct node by using the Connect:Direct bridge. To use a Connect:Direct node as the destination of the multiple file transfer, specify the Connect:Direct bridge agent as the destination agent and specify the destination directory in the form *connect_direct_node_name:directory_path*.

Before you begin

Before transferring files, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. See [“Configuring the Connect:Direct bridge” on page 135](#).

About this task

In this example, the source agent is called FTE_AGENT. The Connect:Direct bridge agent is called CD_BRIDGE. The destination Connect:Direct node is called CD_NODE1. The files to be transferred are /home/jack/data.log, /logs/log1.txt, and /results/latest on the system where FTE_AGENT is located. The files are transferred to the directory /in/files on the system where CD_NODE1 is running.

Procedure

Use the fteCreateTransfer command with the value for the **-dd** (destination directory) parameter in the form *connect_direct_node_name:directory_path*. Specify the value of the **-da** (destination agent) parameter as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by `connect_direct_node_name` is the node that you want the files to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                  -dd CD_NODE1:/in/files /home/jack/data.log
                  /logs/log1.txt /results/latest
```

For more information, see [“fteCreateTransfer: start a new file transfer” on page 417](#).

Results

The source agent FTE_AGENT transfers the first file to the Connect:Direct bridge agent CD_BRIDGE. The Connect:Direct bridge agent temporarily stores the file in the location defined by the `cdTmpDir` property. When the file has been completely transferred from the source agent to the Connect:Direct bridge, the Connect:Direct bridge agent sends the file to the Connect:Direct node that is defined by the `cdNode` agent property. This node sends the file to the destination Connect:Direct node CD_NODE1. The Connect:Direct bridge agent deletes the file from the temporary location when the transfer between the two Connect:Direct nodes is complete. This process is repeated for each specified source file.

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related tasks

[“Transferring a file to a Connect:Direct node” on page 221](#)

You can transfer a file from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge. Specify a Connect:Direct node as the destination of the transfer by specifying the Connect:Direct bridge agent as the destination agent and specifying the destination file in the form `connect_direct_node_name:file_path`.

[“Transferring multiple files to Connect:Direct by using wildcards” on page 226](#)

To transfer multiple files from a Managed File Transfer agent to a Connect:Direct node, use the Connect:Direct bridge. You can use wildcard characters in the source specification that you provide to the **fteCreateTransfer** command. As with all Managed File Transfer transfers involving wildcards, only the last part of the file path can contain a wildcard character. For example, `/abc/def*` is a valid file path and `/abc*/def` is not valid.

[“Transferring a file from a Connect:Direct node” on page 222](#)

You can transfer a file from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form `connect_direct_node_name:file_path`.

[“Transferring multiple files from a Connect:Direct node” on page 225](#)

You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form `connect_direct_node_name:file_path`.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring multiple files from a Connect:Direct node

You can transfer multiple files from a Connect:Direct node to a Managed File Transfer Agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the multiple file transfer

by specifying the Connect:Direct bridge agent as the source agent and specifying one or more source specifications in the form `connect_direct_node_name:file_path`.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. See [“Configuring the Connect:Direct bridge” on page 135](#).

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The destination agent is called FTE_Z, and is running on a z/OS system. The source Connect:Direct node is called CD_NODE1. The files to be transferred are located at the file paths `/in/file1`, `/in/file2`, and `/in/file3` on the system where CD_NODE1 is located. The files are transferred to the partitioned data set `//OBJECT.LIB` on the system where FTE_Z is running.

Procedure

Use the `fteCreateTransfer` command with the values for the source specifications in the form `connect_direct_node_name:file_path` and the value of the `-sa` parameter specified as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by `connect_direct_node_name` is the node that you want the files to be transferred from, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_Z
                  -dp '//OBJECT.LIB' CD_NODE1:/in/file1
                  CD_NODE1:/in/file2 CD_NODE1:/in/file3
```

For more information, see [“fteCreateTransfer: start a new file transfer” on page 417](#).

Results

The Connect:Direct bridge agent CD_BRIDGE requests the first file from the Connect:Direct node CD_NODE1. The Connect:Direct node sends the file to the Connect:Direct bridge. While the file is being transferred from the Connect:Direct node, the Connect:Direct bridge stores the file temporarily in the location defined by the `cdTmpDir` agent property. When the file has finished transferring from the Connect:Direct node to the Connect:Direct bridge, the Connect:Direct bridge sends the file to the destination agent FTE_Z and then deletes the file from the temporary location. This process is repeated for each specified source file.

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring multiple files to Connect:Direct by using wildcards

To transfer multiple files from a Managed File Transfer agent to a Connect:Direct node, use the Connect:Direct bridge. You can use wildcard characters in the source specification that you provide to the `fteCreateTransfer` command. As with all Managed File Transfer transfers involving wildcards, only the last part of the file path can contain a wildcard character. For example, `/abc/def*` is a valid file path and `/abc*/def` is not valid.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of Managed File Transfer. For more information, see [“Configuring the Connect:Direct bridge”](#) on page 135.

About this task

In this example, the source agent is called FTE_AGENT and the Connect:Direct bridge agent is called CD_BRIDGE. The destination Connect:Direct node is called CD_NODE1. The files to be transferred are located in the directory /reports on the system where FTE_AGENT is located. Only files with names that start with report, followed by two characters and the suffix .log, are transferred. For example, the file /reports/report01.log is transferred, but the file /reports/report1.log is not transferred. The files are transferred to the directory /home/fred on the system where CD_NODE1 is running.

Procedure

1. Use the fteCreateTransfer command with the value for the **-dd** (destination directory) parameter in the form *connect_direct_node_name:directory_path*. For the **-da** (destination agent) parameter, specify the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the files to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE  
                 -dd CD_NODE1:/home/fred "/reports/report?? .log"
```

For more information, see [“fteCreateTransfer: start a new file transfer”](#) on page 417.

2. The source agent FTE_AGENT transfers the first file that matches the pattern /reports/report?? .log to the Connect:Direct bridge agent CD_BRIDGE. The Connect:Direct bridge agent temporarily stores the file in the location defined by the cdTmpDir property. When the file has been completely transferred from the source agent to the Connect:Direct bridge, the Connect:Direct bridge agent sends the file to the Connect:Direct node that is defined by the cdNode agent property. This node sends the file to the destination Connect:Direct node CD_NODE1. The Connect:Direct bridge agent deletes the file from the temporary location when the transfer between the two Connect:Direct nodes is complete. This process is repeated for each source file that matches the wildcard pattern /reports/report?? .log.

Note: The list of files that match the pattern /reports/report?? .log varies depending on the operating system of the system where the source agent FTE_AGENT is located.

- If the source agent is located on a system with a Windows operating system, the pattern matching is not case sensitive. The pattern matches all files in the /reports directory with a file name of the form report followed by two characters and a suffix of .log, regardless of the case that the letters are in. For example, Report99 .Log is a match.
- If the source agent is located on a system with a Linux or UNIX operating system, the pattern matching is case sensitive. The pattern matches only those files in the /reports directory with a file name of the form report followed by two characters and a suffix of .log. For example, reportAB .log is a match, but reportAB .LOG and Report99 .Log are not matches.

Recovery and restart for transfers to and from Connect:Direct nodes

Managed File Transfer might be unable to connect to your IBM Sterling Connect:Direct node during a transfer; for example, if the node becomes unavailable. Either Managed File Transfer attempts to recover the transfer, or the transfer fails and an error message is produced.

If the Connect:Direct node becomes unavailable

If the Connect:Direct node becomes unavailable; for example, due to a network or power outage, Managed File Transfer recovers a file transfer in the following ways:

- If Managed File Transfer has not previously successfully connected to the Connect:Direct node as part of this transfer request, the transfer is tried again for a length of time determined by the values of the **cdMaxConnectionRetries** and **recoverableTransferRetryInterval** properties. These properties are specified in the `agent.properties` file for the Connect:Direct bridge agent. The transfer fails, and an error message is produced, after the number of failed attempts reaches the value of the **cdMaxConnectionRetries** property. By default, the transfer is attempted indefinitely, with 60 seconds between attempts.
- If Managed File Transfer has previously successfully connected to the Connect:Direct node as part of this transfer request, the transfer is tried again for a length of time determined by the values of the **cdMaxPartialWorkConnectionRetries** and **recoverableTransferRetryInterval** properties. The transfer fails, and an error message is produced, after the number of failed attempts reaches the value of the **cdMaxPartialWorkConnectionRetries** property. By default, the transfer is attempted indefinitely, with 60 seconds between attempts.
- For certain types of Connect:Direct node failure, for example the node being forcibly stopped, Connect:Direct processes go into Held Due to Error (HE) status when the node recovers. After the node recovers, Managed File Transfer automatically resumes any Connect:Direct processes that are related to the file transfer and have a status of HE.
- If the transfer fails, any temporary files relating to the transfer are deleted from the system that hosts the Connect:Direct bridge. The location of these temporary files is defined by the **cdTmpDir** property.
- If the transfer is from Managed File Transfer to Connect:Direct, and a source disposition of delete is specified, then the source files are not deleted if the transfer fails.

If the Connect:Direct node user credentials are invalid

If Managed File Transfer fails to connect to the Connect:Direct node because the credentials of the user are rejected by the node, the transfer fails and an error message is produced. In this situation, check that you have provided the correct user credentials for the Connect:Direct node. For more information, see [“Mapping credentials for Connect:Direct” on page 138](#).

If the Connect:Direct bridge agent becomes unavailable

If the Connect:Direct bridge agent becomes unavailable, any ongoing file transfers recover in the same way as standard Managed File Transfer transfers. For more information, see [“MFT recovery and restart” on page 235](#).

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

[“MFT recovery and restart” on page 235](#)

If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, Managed File Transfer recovers as follows in these scenarios:

Related tasks

[“Configuring the Connect:Direct bridge” on page 135](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Submitting a user-defined Connect:Direct process from a file transfer request

You can submit a transfer request for a transfer that goes through the Connect:Direct bridge agent that calls a user-defined Connect:Direct process as part of the file transfer.

By default, when you submit a file transfer request for a transfer that goes through the Connect:Direct bridge, the Connect:Direct bridge agent generates the Connect:Direct process that is used to transfer the file to or from the remote Connect:Direct node.

However, you can configure the Connect:Direct bridge agent to instead call a user-defined Connect:Direct process by using the `ConnectDirectProcessDefinition.xml` file.

The ConnectDirectProcessDefinition.xml file

The `fteCreateCDAgent` command creates the file `ConnectDirectProcessDefinitions.xml` in the agent configuration directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name`. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

The file defines one or more process sets that includes the location of one or more Connect:Direct processes that are called as part of a transfer. Each process set includes a number of conditions. If the transfer satisfies all of the conditions of the process set, the process set is used to specify which Connect:Direct processes are called by the transfer. For more information, see [“Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file” on page 143.](#)

Intrinsic symbolic variables

You can use the intrinsic symbolic variables that are defined by Managed File Transfer to substitute values into user-defined Connect:Direct processes. To follow the Connect:Direct naming convention, all intrinsic symbolic variables used by Managed File Transfer have the format `%FTE` followed by five uppercase alphanumeric characters.

When creating a process to transfer files from a Connect:Direct node to the Connect:Direct bridge system, you must use the intrinsic variable `%FTETFILE` as the value of TO FILE in the Connect:Direct process. When creating a process to transfer files to a Connect:Direct node from the Connect:Direct bridge system, you must use the intrinsic variable `%FTEFFILE` as the value of FROM FILE in the Connect:Direct process. These variables contain the temporary file paths that the Connect:Direct bridge agent uses for transfers into and out of the Managed File Transfer network.

For more information about intrinsic symbolic variables, see the Connect:Direct product documentation.

Sample Connect:Direct processes

Managed File Transfer provides sample Connect:Direct processes. These samples are located in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/ConnectDirectProcessTemplates`.

Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file

Specify which Connect:Direct process to start as part of a Managed File Transfer transfer. Managed File Transfer provides an XML file that you can edit to specify process definitions.

About this task

The `fteCreateCDAgent` command creates the file `ConnectDirectProcessDefinitions.xml` in the agent configuration directory `MQ_DATA_PATH/mqft/config/coordination_queue_manager/`

`agents/cd_bridge_agent_name`. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

For each process that you want to specify to call as part of a transfer through the Connect:Direct bridge, perform the following steps:

Procedure

1. Define the Connect:Direct process that you want the Connect:Direct bridge agent to call as part of the transfer and save the process template in a file.
2. Open the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name/ConnectDirectProcessDefinitions.xml` file in a text editor.
3. Create a `<processSet>` element.
4. Inside the `<processSet>` element, create a `<condition>` element.
5. Inside the `<condition>` element, create one or more elements that define a condition that the transfer request must match to call the Connect:Direct process you defined in Step 1. These elements can be either `<match>` elements or `<defined>` elements.
 - Use a `<match>` element to specify that the value of a variable must match a pattern. Create the `<match>` element with the following attributes:
 - `variable` - the name of the variable whose value is compared. The variable is an intrinsic symbol. For more information, see [“Substitution variables for use with user-defined Connect:Direct processes”](#) on page 633.
 - `value` - the pattern to compare to the value of the specified variable.
 - Optional: `pattern` - the type of pattern used by the value of the `value` attribute. This pattern type can be `wildcard` or `regex`. This attribute is optional and the default is `wildcard`.
 - Use a `<defined>` element to specify that a variable must have a value defined. Create the `<defined>` element with the following attribute:
 - `variable` - the name of the variable that must have a value defined. The variable is an intrinsic symbol. For more information, see [“Substitution variables for use with user-defined Connect:Direct processes”](#) on page 633.

The conditions specified within the `<condition>` element are combined with a logical AND. All conditions must be met for the Connect:Direct bridge agent to call the process specified by this `<processSet>` element. If you do not specify a `<condition>` element, the process set matches all transfers.

6. Inside the `<processSet>` element, create a `<process>` element.
7. Inside the `<process>` element, create a `<transfer>` element.

The transfer element specifies the Connect:Direct process that the Connect:Direct bridge agent calls as part of the transfer. Create the `<transfer>` element with the following attribute:

- `process` - the location of the Connect:Direct process that you defined in step 1. The location of this file is specified with an absolute path or relative to the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent_name` directory.

Results

When searching for a condition match, the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks

[“Configuring the Connect:Direct bridge”](#) on page 135

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a

Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[“Connect:Direct process definitions file format” on page 565](#)

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 384](#)

The `fteCreateCDAgent` command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

Using intrinsic symbolic variables in Connect:Direct processes that are called by Managed File Transfer

You can call a user-defined Connect:Direct process from a Managed File Transfer transfer and pass in information from the transfer to the Connect:Direct process by using intrinsic symbolic variables in the process definition.

About this task

This example uses intrinsic symbolic variables to pass information from a Managed File Transfer transfer in to a user-defined Connect:Direct process. For more information about intrinsic symbolic variables used by Managed File Transfer, see [“Substitution variables for use with user-defined Connect:Direct processes” on page 633](#).

In this example, the file is transferred from a Managed File Transfer Agent to a Connect:Direct bridge node. The first part of the transfer is performed by Managed File Transfer. The second part of the transfer is performed by a user-defined Connect:Direct process.

Procedure

1. Create a Connect:Direct process that uses intrinsic symbolic variables.

```
%FTEPNAME PROCESS
SNODE=%FTESNODE
PNODEID=(%FTEPUSER,%FTEPPASS)
SNODEID=(%FTESUSER,%FTESPASS)

COPY001 COPY
FROM (
  FILE=%FTEFFILE
  DISP=%FTEFDISP
)
TO (
  FILE=%FTETFILE
  DISP=%FTETDISP
)
PEND
```

2. Save this process to a text file at the following location: `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/cd_bridge_agent/Example.cdp`
3. Edit the `ConnectDirectProcessDefinition.xml` file to include a rule that calls the Connect:Direct process that you created in Step 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/
  ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="TOBERMORY" pattern="wildcard"/>
    </tns:condition>
    <tns:process>
      <tns:transfer process="Example.cdp"/>
    </tns:process>
  </tns:processSet>
</tns:cdprocess>
```

```
</tns:processSet>
```

```
</tns:cdprocess>
```

In this example, if a transfer request is submitted to the Connect:Direct bridge agent that has TOBERMORY as its source or destination Connect:Direct node, the Example.cdp Connect:Direct process is called.

4. Submit a file transfer request that satisfies the conditions that you defined in the ConnectDirectProcessDefinition.xml file in Step 3.

For example,

```
fteCreateTransfer -sa ORINOCO -da CD_BRIDGE
                 -sm QM_WIMBLEDON -dm QM_COMMON
                 -de overwrite -df TOBERMORY:/home/bulgaria/destination.txt
                 -sd leave c:\bungo\source.txt
```

In this example, the destination Connect:Direct node is TOBERMORY. This node is the secondary node in the transfer and the value of %FTESNODE is set to TOBERMORY. This command matches the condition that is set in the ConnectDirectProcessDefinition.xml file.

5. Managed File Transfer transfers the source file to a temporary location on the same system as the Connect:Direct bridge agent.
6. The Connect:Direct bridge agent sets the values of the intrinsic symbolic variables from the information in the transfer request and configuration information.

The intrinsic symbolic variables are set to the following values:

- %FTEPNAME=*process_name* - This value is an 8 character process name generated by the Connect:Direct bridge agent.
 - %FTESNODE=TOBERMORY - This value is set from the **-df** parameter of the **fteCreateTransfer** command.
 - %FTEPUSER=*primary_node_user* - This information is taken from the ConnectDirectCredentials.xml file.
 - %FTEPPASS=*primary_node_user_password* - This information is taken from the ConnectDirectCredentials.xml file.
 - %FTESUSER=*secondary_node_user* - This information is taken from the ConnectDirectCredentials.xml file.
 - %FTESPASS=*secondary_node_user_password* - This information is taken from the ConnectDirectCredentials.xml file.
 - %FTEFFILE=*temporary_location* - This value is the temporary location of the file on the same system as the Connect:Direct bridge agent.
 - %FTEFDISP=leave - This value is set from the **-sd** parameter of the **fteCreateTransfer** command.
 - %FTETFILE=/home/bulgaria/destination.txt - This value is set from the **-df** parameter of the **fteCreateTransfer** command.
 - %FTETDISP=overwrite - This value is set from the **-de** parameter of the **fteCreateTransfer** command.
7. The Connect:Direct process is started on the Connect:Direct bridge node. Connect:Direct transfers the file from the temporary location on the Connect:Direct bridge system to the destination /home/bulgaria/destination.txt on the system where the Connect:Direct node TOBERMORY is running.

Related concepts

[“Submitting a user-defined Connect:Direct process from a file transfer request” on page 229](#)

You can submit a transfer request for a transfer that goes through the Connect:Direct bridge agent that calls a user-defined Connect:Direct process as part of the file transfer.

Related reference

[“Substitution variables for use with user-defined Connect:Direct processes” on page 633](#)

You can define values to substitute in to user-defined Connect:Direct processes by using intrinsic symbolic variables that are specific to Managed File Transfer.

Using Connect:Direct processes to submit Managed File Transfer transfer requests

You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

Managed File Transfer provides the following commands for use with Connect:Direct processes:

ftetag

Specify this command in a step that precedes the **ftebxfer** or **ftecxfer** command to create the required audit information for the transfer. This command takes the source specification of the transfer as a parameter. For information about the format of source specification, see [“fteCreateTransfer: start a new file transfer” on page 417](#).

ftebxfer

Specify this command to create a file transfer request when the queue manager that the transfer request is submitted to is located on the same system as the Connect:Direct node that submits the command. This command takes the same parameters as the **fteCreateTransfer** command. For information about these parameters, see [“fteCreateTransfer: start a new file transfer” on page 417](#). This command also has an additional parameter:

-qmgrname

Required. The name of the queue manager to submit the command to.

ftecxfer

Specify this command to create a file transfer request when the queue manager that the transfer request is submitted to is located on a different system to the Connect:Direct node that submits the command. This command takes the same parameters as the **fteCreateTransfer** command. For information about the parameters, see [“fteCreateTransfer: start a new file transfer” on page 417](#). This command also has three additional parameters:

-qmgrname

Required. The name of the queue manager to submit the command to.

-connname

Required. The host and port of the queue manager to submit the command to, specified in IBM MQ CONNAME format. For example, `host.example.com(1337)`.

-channelname

Optional. The name of the channel to use to connect to the queue manager to submit the command to. If this is not specified, a default of `SYSTEM.DEF.SVRCONN` is used.

Related tasks

[“Creating and submitting a Connect:Direct process that calls Managed File Transfer by using the Connect:Direct Requester” on page 234](#)

The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls Managed File Transfer.

Related reference

[“Example: A Connect:Direct process file that calls MFT commands” on page 637](#)

An example Connect:Direct process file that calls the Managed File Transfer **ftetag** command and the **ftecxfer** command.

Creating and submitting a Connect:Direct process that calls Managed File Transfer by using the Connect:Direct Requester

The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls Managed File Transfer.

About this task

This task describes how to create a Connect:Direct process that calls the Managed File Transfer **ftecxfer** command or the **ftebxfer** command. Use the **ftecxfer** command when the queue manager that the transfer request is submitted to is located on a different system to the Connect:Direct node that submits the command. Use the **ftebxfer** command when the queue manager that the transfer request is submitted to is located on the same system as the Connect:Direct node that submits the command. The **ftecxfer** command makes a client connection to the agent queue manager of the source agent of the transfer. Before calling the **ftecxfer** command, you must call the **ftetag** command and pass it the source specification information. This allows the process to be logged and audited in the same way as transfers initiated from Managed File Transfer.

Procedure

1. Start the Connect:Direct Requester.
2. In the **Nodes** tab of the panel, select the Connect:Direct node that is used as the primary node of the process.
3. Select **File > New > Process**. The **Process Properties** window opens.
4. In the **Name:** field, type the name of the process.
5. Select the secondary node from the **Snode > Name:** list.
6. Select the operating system of the secondary node from the **Snode > Operating System:** list.
7. Optional: Complete any further information in this window that you require.
8. Click **OK**. The **Process Properties** window closes.
9. Create a statement that runs the Managed File Transfer **ftetag** command.
 - a) Right-click in the **Process** window on the **End** statement.
 - b) Select **Insert > Run Task**. The **Run Task Statement** window opens.
 - c) In the **Label:** field, type Tag.
 - d) In the **Optional Parameters or Commands** field, type `pgm(MQ_INSTALLATION_PATH/bin/ftetag) args(source_specification)`. For more information about the format of *source_specification*, see [“fteCreateTransfer: start a new file transfer” on page 417](#).
 - e) Click **OK**. The **Run Task Statement** window closes.
10. Create a statement that runs the Managed File Transfer **ftecxfer** or **ftebxfer** command.
 - a) Right-click in the **Process** window on the **End** statement.
 - b) Select **Insert > Run Task**. The **Run Task Statement** window opens.
 - c) In the **Label:** field, type Transfer.
 - d) In the **Optional Parameters or Commands** field, type `pgm(MQ_INSTALLATION_PATH/bin/ftecxfer) args(parameters)` or `pgm(MQ_INSTALLATION_PATH/bin/ftebxfer) args(parameters)` depending on which command you choose. The parameters used by the **ftecxfer** and **ftebxfer** commands are the same as the parameters used by the **fteCreateTransfer** command, plus some additional parameters specific to **ftecxfer** and **ftebxfer**. For more information, see [“fteCreateTransfer: start a new file transfer” on page 417](#) and [“Using Connect:Direct processes to submit Managed File Transfer transfer requests” on page 233](#).
 - e) Click **OK**. The **Run Task Statement** window closes.

11. Optional: Create any additional statements that you require.
12. Submit the process.
 - a) Right-click in the **Process** window.
 - b) Select **Submit**. The **Connect:Direct Attach** window opens.
 - c) Enter the user name and password to use to run the process.
 - d) Click **OK**.

Related concepts

[“Using Connect:Direct processes to submit Managed File Transfer transfer requests” on page 233](#)

You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

Working with MFT from IBM Integration Bus

You can work with Managed File Transfer from IBM Integration Bus using the FTEOutput and FTEInput nodes.

- Use the FTEInput node to transfer a file across the network using Managed File Transfer and then process that file as part of an Integration Bus flow.
- Use the FTEOutput node to transfer a file that has been output by an Integration Bus flow to another location in the network.

The agents that transfer files to or from the broker agent can be at any level of Managed File Transfer. For more information, refer to the [IBM Integration Bus product documentation](#).

MFT recovery and restart

If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, Managed File Transfer recovers as follows in these scenarios:

- Typically, if there is a problem while a file is being transferred, Managed File Transfer recovers and restarts that file transfer after the problem is repaired.
- If a file that was in the process of being transferred is deleted or changed while the agent or queue manager are unavailable, the transfer fails and you get a message in the transfer log that provides details about the failure.
- If an agent process fails during a file transfer, the transfer continues when you restart the agent.
- If an agent loses the connection to its agent queue manager, the agent waits while trying to reconnect to the queue manager. When the agent successfully reconnects to its queue manager, the current transfer continues.
- If the agent is stopped for any reason, any resource monitors associated with an agent stop polling. When the agent recovers, the monitors are also restarted, and resource polling resumes.
- For a file transfer with a source disposition of `delete`, if a recovery occurs after all the data is sent from a source agent to a destination agent, the source file is unlocked before deletion. This unlocking means that the source file could possibly be modified before the file is deleted. Therefore, it is considered to be unsafe to delete the source file and the following warning is displayed:

```
BFGTR0075W: The source file has not been deleted because it is possible that the source file was modified after the source file was transferred.
```

In this case, verify that the content of the source file is unmodified and then manually delete the source file.

You can check the status of your transfers in the IBM MQ Explorer. If any transfers appear as **Stalled**, you might need to take corrective action because the stalled status denotes an issue either with the agent or between the two agents involved in the transfer.

Related concepts

[“Timeout option for file transfers in recovery” on page 236](#)

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

V 9.0.1 Timeout option for file transfers in recovery

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

Before IBM MQ 9.0.1, the default behavior of a Managed File Transfer source agent is to keep trying to recover a stalled transfer until it is successful. Because the new parameter is optional, you do not have to set it and the transfers follow the default behavior.

From IBM MQ 9.0.1, you can specify one of the following options:

-1

The agent continues to attempt to recover the stalled transfer until the transfer is successful. Using this option is the equivalent of the default behavior of the agent when the property is not set.

0

The agent stops the file transfer as soon as it enters recovery.

>0

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. A value of 21600 indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this parameter is 999999999.

Setting the recovery retry timeout for all the transfers for one source agent

To set a recovery timeout that applies to all the transfers for a source agent, add the parameter and value pair to the [agent.properties](#) file.

In this example, setting a transfer recovery timeout value as -1 means that the agent continues to try to recover the stalled transfer until it completes successfully:

```
transferRecoveryTimeout=-1
```

Setting a transfer recovery timeout value as 0 means the agent marks transfer as failed immediately upon entering recovery:

```
transferRecoveryTimeout=0
```

Setting a transfer recovery timeout value of 21600 means the agent keeps retrying a stalled transfer for 6 hours before the transfer is marked as failed:

```
transferRecoveryTimeout=21600
```

Setting or overriding the recovery retry timeout for individual transfers

You can set the recovery retry timeout parameter for an individual transfer when you are:

- Creating a transfer by using the [fteCreateTransfer](#) command in Managed File Transfer command line interface or IBM MQ Explorer
- Creating a transfer template by using the [“fteCreateTemplate: create new file transfer template” on page 401](#) command in Managed File Transfer command line interface or IBM MQ Explorer
- Creating a monitor by using the [fteCreateMonitor](#) Managed File Transfer command line interface or IBM MQ Explorer

- Copying or moving files by using `fte:filecopy` or `fte:filemove` Ant tasks

Setting the `-rt` value to `-1` for a transfer is the equivalent of the default behavior, recovery continues until the transfer is successful or it is manually canceled by the user, for example

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt -1 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

For more information about using the transfer recovery timeout parameter while creating a new transfer, see [fteCreateTransfer command](#).

Setting the `-rt` parameter value to `0` indicates that if the transfer initiated by using this template is stalled, it fails immediately and no recovery is attempted, for example

```
fteCreateTemplate -tn "payroll accounts monthly report template" -rt 0 -sa PAYROLL -sm
QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

For more information about using the transfer recovery timeout parameter while creating a new transfer template, see [fteCreateTemplate command](#).

You can use IBM MQ Explorer to set the recovery timeout parameter and value for transfers. For more information on using IBM MQ Explorer to configure transfers, see [“Starting a new file transfer” on page 152](#) and [“Creating a file transfer template using IBM MQ Explorer” on page 188](#).

You can also set the recovery timeout by including the **transferRecoveryTimeout** option and value, with the **fte:filecopy** or **fte:filemove** elements for moving or copying files by using Ant tasks, for example

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
rcproperty="copy.result" transferRecoveryTimeout="0">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
</fte:filecopy>
```

When the file copy task is initiated and the transfer enters recovery, the transfer stops immediately without attempting recovery. Setting the **transferRecoveryTimeout** option with **fte:filecopy** or **fte:filemove**, overrides the value set in `agent.properties` file. If the **transferRecoveryTimeout** value is not set with **fte:filecopy** or **fte:filemove**, the value of **transferRecoveryTimeout** parameter from the `agent.properties` file is used. For more information, see [“fte:filecopy Ant task” on page 826](#) and [“fte:filemove Ant task” on page 830](#).

Handling recovery timeout precedence

Transfer recovery timeout value as specified through the command line interface argument for create transfer, template, or monitor commands (including setting the option in IBM MQ Explorer Wizard) or as specified in the **fte:filespec** nested element, takes precedence over the value that is specified for the **transferRecoveryTimeout** parameter in the `agent.properties` file for the source agent. For example, for the command

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:\import\transferredfile.txt
C:\export\originalfile.txt
```

that is started without the `-rt` parameter and value pair, the source agent AGENT1 checks the `agent.properties` file for a **transferRecoveryTimeout** value to determine the recovery timeout behavior.

If in the `agent.properties` file, the **transferRecoveryTimeout** is not set or is set to `-1`, the agent follows the default behavior and tries to recover the transfer until it is successful.

When the recovery timeout option **-rt** is specified through the Managed File Transfer command line interface, for example, with the **fteCreateTransfer** command, this value takes precedence over the value in the `agent.properties` file and is used as the setting for the transfer.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -rt 21600 -df C:\import\transferredfile.txt  
C:\export\originalfile.txt
```

Handling recovery timeout counter

The recovery timeout counter starts when the transfer enters recovering state. A transfer log message is published to the `SYSTEM.FTE` topic with the topic string `Log/agent_name/transfer_ID` to indicate that the transfer status is changed to recovering and the source agent clock time at which the status changed. If the transfer is resumed within the set retry interval and does not reach the recovery timeout (`counter<=recovery timeout`), then the counter is reset to 0, ready to start again if the transfer enters recovery.

If the counter reaches the maximum value set for the recovery timeout (`counter==recovery timeout`), the recovery of the transfer stops and the source agent reports the transfer as failed. This type of transfer failure, caused by the fact that the transfer reached the recovery timeout, is indicated by a new message code, `RECOVERY_TIMEOUT (69)`. Another transfer log message is published to the `SYSTEM.FTE` topic, with a topic string of `Log/agent_name/transfer_ID`, to indicate that the transfer is failed and includes a new message, the new return code, and the source agent's event log. Source Agent's event log is updated with a message when any of the following events occur during recovery:

- When the recovery timeout parameter is set to a value greater than `-1`, the Managed File Transfer enters recovery. The agent's event log is updated to indicate the start of the recovery timer for the **TransferId** and the amount of time the source agent waits before it initiates the recovery timeout processing.
- When the recovering Managed File Transfer is resumed, the source agent's event log is updated with a new message to indicate that the **TransferId** that was in recovery is resumed.
- When a recovering Managed File Transfer has timed out, the source agent's event log is updated to indicate the **TransferId** that failed while recovering, due to recovery timeout.

These log messages enable the users (subscribers and loggers) to identify the transfers that failed due to the transfer recovery timeout.

The counter for the recovery timeout is always at the source agent. However, if the destination agent fails to receive information from the source agent in a timely manner, it can send a request to the source agent to put the transfer in recovery. For a transfer where the recovery timeout option is set, the source agent starts the recovery timeout counter when it receives the request from the destination agent.

Manual handling is still required for transfers that do not use the recovery timeout option, the failed, and partially complete transfers.

For transfer sets, where a single transfer request is issued for multiple files, and some of the files completed successfully but one completed only partially, the transfer is still marked as failed as it did not complete as expected. The source agent might have timed out while transferring the partially completed file.

Ensure that the destination agent and file server are ready and in a state to accept file transfers.

You have to issue the transfer request again for the entire set, but to avoid problems because some of the files remain on the destination from the initial transfer attempt, you can issue the new request with the `overwrite if existing` option specified. This ensures that the incomplete set of files from the previous transfer attempt are cleaned up as a part of the new transfer, before the files are written to the destination again.

Traces and messages

Tracing points are included for diagnostic purposes. Recovery timeout value, start of the retry interval, start of the resume period and counter reset, and whether the transfer timed out and failed, are logged. In

case of a problem or unexpected behavior, you can collect the source agent output log and trace files, and provide them when requested by IBM support, to help with troubleshooting.

Messages notify the user when a transfer enters recovery (BFGTR0081I), is terminated because it timed out from recovery (BFGSS0081E) and when it resumes after being in recovery (BFGTR0082I).

BFGTR0001 - BFGTR9999

BFGTR0081I

BFGTR0082I

BFGSS0001 - BFGSS9999

BFGSS0081E

Developing applications for MFT

Specifying programs to run with MFT

You can run programs on a system where a Managed File Transfer Agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

There are five scenarios in which you can specify a program to run:

- As part of a transfer request, at the source agent, before the transfer starts
- As part of a transfer request, at the destination agent, before the transfer starts
- As part of a transfer request, at the source agent, after the transfer completes
- As part of a transfer request, at the destination agent, after the transfer completes
- Not as part of a transfer request. You can submit a request to an agent to run a program. This scenario is sometimes referred to as a managed call.

User exits and program calls are invoked in the following order:

```
- SourceTransferStartExit(onSourceTransferStart).  
- PRE_SOURCE Command.  
- DestinationTransferStartExits(onDestinationTransferStart).  
- PRE_DESTINATION Command.  
- The Transfer request is performed.  
- DestinationTransferEndExits(onDestinationTransferEnd).  
- POST_DESTINATION Command.  
- SourceTransferEndExits(onSourceTransferEnd).  
- POST_SOURCE Command.
```

Notes:

1. The **DestinationTransferEndExits** is run only when the transfer completes, either successfully or partially successfully.
2. The **postDestinationCall** is run only when the transfer completes, either successfully or partially successfully.
3. The **SourceTransferEndExits** is run for successful, partially successful, or failed transfers.
4. The **postSourceCall** is called only if:
 - The transfer was not canceled.
 - There is a successful or partially successful outcome.
 - Any post-destination transfer programs ran successfully.

There are several ways to specify a program that you want to run. These options are as follows:

Use an Apache Ant task

Use one of the `fte:filecopy`, `fte:filemove`, and `fte:call` Ant tasks to start a program. Using an Ant task, you can specify a program in any of the five scenarios, using the `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc`, and `fte:command` nested elements. For more information, see [“Program invocation nested elements” on page 842](#).

Edit the file transfer request message

You can edit the XML that is generated by a transfer request. Using this method, you can run a program in any of the five scenarios, by adding **preSourceCall**, **postSourceCall**, **preDestinationCall**, **postDestinationCall**, and **managedCall** elements to the XML file. Then, use this modified XML file as the transfer definition for a new file transfer request, for example with the **fteCreateTransfer -td** parameter. For more information, see [“MFT agent call request message examples” on page 770](#).

Use the fteCreateTransfer command

You can use the **fteCreateTransfer** command to specify programs to start. You can use the command to specify programs to run in the first four scenarios, as part of a transfer request, but you cannot start a managed call. For information about the parameters to use, see [“fteCreateTransfer: start a new file transfer” on page 417](#). For examples of using this command, see [“Examples of using fteCreateTransfer to start programs” on page 818](#).

Related reference

[“commandPath MFT property” on page 347](#)

Use the `commandPath` property to specify the locations that Managed File Transfer can run commands from. Take extreme care when you set this property because any command in one of the specified `commandPaths` can effectively be called from a remote client system that is able to send commands to the agent.

Managed calls

Managed File Transfer (MFT) agents are typically used to transfer files or messages. These are known as *Managed Transfers*. Agents can also be used to run commands, scripts or JCL without the need for transferring files or messages. This capability is known as *Managed Calls*.

Managed call requests can be submitted to an agent in several ways:

- Using the `fte:call` Ant task.
- Configuring a resource monitor with a task XML that runs a command or a script. See [Configuring monitor tasks to start commands and scripts](#) for more information.
- Directly putting an XML message into the agent's command queue. See [File transfer request message format](#) for more details on the Managed Call XML schema.

For managed calls, the directory containing the command or script that is being run must be specified in the agent property **commandPath**.

Managed calls cannot run commands or scripts that are located in directories which are not specified in the agent's **commandPath**. This is to ensure that the agent does not run any malicious code.

Additionally, you can also enable authority checking on an agent to ensure that only authorized users are allowed to submit managed call requests. For more information on this, see [Restricting user authorities on MFT agent actions](#).

The command, script, or JCL invoked as part of a managed call runs as an external process, which is monitored by the agent. When the process exits, the managed call completes and the return code from the process is made available to either the agent or the script that invoked the **fte: Ant call** Ant task.

If the managed call was started by the **fte:call** Ant task, then your Ant script can check the value of the return code to determine whether the managed call was successful or not.

For all other types of managed calls, you can specify which return code values should be used to indicate that the managed call completed successfully. The agent compares the return code from the process against these return codes when the external process finishes.

Note: Because managed calls run as external processes, they cannot be canceled once they have started.

Managed calls and source transfer slots

An agent contains a number of source transfer slots, as specified by the agent property **maxSourceTransfers**, described in [Advanced agent properties: Transfer limit](#).

Whenever a managed call or a managed transfer is run, they occupy a source transfer slot. The slot is released when the managed call or managed transfer completes.

If all the source transfer slots are in use when an agent receives either a new managed call or managed transfer request, the request is queued by the agent until a slot becomes available.

If a managed call starts a managed transfer (for example, if a managed call runs an Ant script and that Ant script uses the [fte:filecopy](#) or [fte:filemove](#) task to transfer a file), then two source transfer slots are required:

- One for the managed transfer
- One for the managed call

In this situation, it is important to note that if the managed transfer either takes a long time to complete, or goes into recovery, then the two source transfer slots are occupied until either the managed transfer completes or is canceled

Because of this, you should consider the design of a managed call to ensure that it does not occupy source transfer slots for a long period of time.

Using Apache Ant with MFT

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

You can use the **fteAnt** command to run Ant tasks in a Managed File Transfer environment that you have already configured. You can use file transfer Ant tasks from your Ant scripts to coordinate complex file transfer operations from an interpreted scripting language.

For more information about Apache Ant, see the Apache Ant project web page: <https://ant.apache.org/>

Related concepts

[“Getting started using Ant scripts with MFT” on page 242](#)

Using Ant scripts with Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

Related reference

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

[“fteAnt: run Ant tasks in MFT” on page 361](#)

The **fteAnt** command runs Ant scripts in an environment that has Managed File Transfer Ant tasks available.

[“Sample Ant tasks for MFT” on page 243](#)

There are a number of sample Ant scripts provided with your installation of Managed File Transfer. These samples are located in the directory `MQ_INSTALLATION_PATH/mqft/samples/fteant`. Each sample

script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

Getting started using Ant scripts with MFT

Using Ant scripts with Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

Ant scripts

Ant scripts (or build files) are XML documents defining one or more targets. These targets contain task elements to run. Managed File Transfer provides tasks which you can use to integrate file transfer function into Apache Ant. To learn about Ant scripts, see the Apache Ant project web page: <https://ant.apache.org/>

Examples of Ant scripts that use Managed File Transfer tasks are provided with your product installation in the directory `MQ_INSTALLATION_PATH/mqft/samples/fteant`

On protocol bridge agents, Ant scripts are run on the protocol bridge agent system. These Ant scripts do not have direct access to the files on the FTP or SFTP server.

Namespace

A namespace is used to differentiate the file transfer Ant tasks from other Ant tasks that might share the same name. You define the namespace in the project tag of your Ant script.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="do_ping">

  <target name="do_ping">
    <fte:ping cmdqm="qm@localhost@1414@SYSTEM.DEF.SVRCONN" agent="agent1@qm1"
      rcproperty="ping.rc" timeout="15"/>
  </target>
</project>
```

The attribute `xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs"` tells Ant to look for the definitions of tasks prefixed by `fte` in the library `com.ibm.wmqfte.ant.taskdefs`.

You do not need to use `fte` as your namespace prefix; you can use any value. The namespace prefix `fte` is used in all examples and sample Ant scripts.

Running Ant scripts

To run Ant scripts that contain the file transfer Ant tasks use the **fteAnt** command. For example:

```
fteAnt -file ant_script_location/ant_script_name
```

For more information, see [“fteAnt: run Ant tasks in MFT” on page 361](#).

Return codes

The file transfer Ant tasks return the same return codes as the Managed File Transfer commands. For more information, see [“Return codes for MFT” on page 322](#).

Related reference

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

[“fteAnt: run Ant tasks in MFT” on page 361](#)

The **fteAnt** command runs Ant scripts in an environment that has Managed File Transfer Ant tasks available.

[“Sample Ant tasks for MFT” on page 243](#)

There are a number of sample Ant scripts provided with your installation of Managed File Transfer. These samples are located in the directory `MQ_INSTALLATION_PATH/mqft/samples/fteant`. Each sample script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

Sample Ant tasks for MFT

There are a number of sample Ant scripts provided with your installation of Managed File Transfer. These samples are located in the directory `MQ_INSTALLATION_PATH/mqft/samples/fteant`. Each sample script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

email

The email sample demonstrates how to use Ant tasks to transfer a file and send an email to a specified email address if the transfer fails. The script checks that the source and destination agents are active and able to process transfers by using the Managed File Transfer `ping` task. If both agents are active, the script uses the Managed File Transfer `filecopy` task to transfer a file between the source and destination agents, without deleting the original file. If the transfer fails the script sends an email containing information about the failure by using the standard Ant email task.

hub

The hub sample is made up of two scripts: `hubcopy.xml` and `hubprocess.xml`. The `hubcopy.xml` script shows how you can use Ant scripting to build 'hub and spoke' style topologies. In this sample, two files are transferred from agents running on spoke machines to an agent running on the hub machine. Both files are transferred at the same time, and when the transfers are complete the `hubprocess.xml` Ant script is run on the hub machine to process the files. If both files transfer correctly, the Ant script concatenates the contents of the files. If the files do not transfer correctly, the Ant script cleans up by deleting any file data that was transferred. For this example to work correctly, you must put the `hubprocess.xml` script on the command path of the hub agent. For more information about setting the command path of an agent, see [commandPath](#).

librarytransfer (IBM i platform only)

IBM i

IBM i The librarytransfer sample demonstrates how to use Ant tasks to transfer an IBM i library on one IBM i system to a second IBM i system.

IBM i

IBM WebSphere MQ File Transfer Edition 7.0.2 on IBM i does not include direct support for transfers of native IBM i library objects. The librarytransfer sample uses the native save file support on IBM i with predefined Ant Tasks available in Managed File Transfer to transfer native library objects between two IBM i systems. The sample uses a `<presrc>` nested element in a Managed File Transfer `filecopy` task to invoke an executable script `librarysave.sh` that saves the requested library on the source agent system into a temporary save file. The save file is moved by the `filecopy` ant task to the destination agent system where a `<postdst>` nested element is used to invoke the executable script `libraryrestore.sh` to restore the library saved in the save file to the destination system.

IBM i

Before you run this sample, you need to complete some configuration as described in the `librarytransfer.xml` file. You must also have a working Managed File Transfer environment on two IBM i machines. The setup must consist of a source agent running on the first IBM i machine and a destination agent running on the second IBM i machine. The two agents must be able to communicate with each other.

IBM i

The librarytransfer sample consists of the following three files:

- `librarytransfer.xml`
- `librarysave.sh` (`<presrc>` executable script)

- `libraryrestore.sh` (<postdst> executable script)

The sample files are located in the following directory: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer`

IBM i

To run this sample the user must complete the following steps:

1. Start a Qshell session. At an IBM i command window type: `STRQSH`
2. Change directory to the `bin` directory as follows:

```
cd /QIBM/ProdData/WMQFTE/V7/bin
```

3. After completing the required configuration, run the sample by using the following command:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer/librarytransfer.xml
```

physicalfiletransfer (IBM i platform only)

IBM i

The `physicalfiletransfer` sample demonstrates how to use Ant tasks to transfer a Source Physical or Database file from a library on one IBM i system to a library on a second IBM i system.

IBM i

IBM WebSphere MQ File Transfer Edition 7.0.2 on IBM i does not include direct support for transfers of native Source Physical or Database files on IBM i. The `physicalfiletransfer` sample uses the native save file support on IBM i with predefined Ant Tasks available in Managed File Transfer to transfer complete Source Physical and Database files between two IBM i systems. The sample uses a <presrc> nested element within a Managed File Transfer filecopy task to invoke an executable script `physicalfilesave.sh` to save the requested Source Physical or Database file from a library on the source agent system into a temporary save file. The save file is moved by the filecopy ant task to the destination agent system where a <postdst> nested element is used to invoke the executable script `physicalfilerestore.sh` then restores the file object inside the save file into a specified library on the destination system.

IBM i

Before you run this sample, you must complete some configuration as described in the `physicalfiletransfer.xml` file. You must also have a working Managed File Transfer environment on two IBM i systems. The setup must consist of a source agent running on the first IBM i system and a destination agent running on the second IBM i system. The two agents must be able to communicate with each other.

IBM i

The `physicalfiletransfer` sample consists of the following three files:

- `physicalfiletransfer.xml`
- `physicalfilesave.sh` (<presrc> executable script)
- `physicalfilerestore.sh` (<postdst> executable script)

The sample files are located in the following directory: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer`

IBM i

To run this sample the user must complete the following steps:

1. Start a Qshell session. At an IBM i command window type: `STRQSH`
2. Change directory to the `bin` directory as follows:

```
cd /QIBM/ProdData/WMQFTE/V7/bin
```

3. After completing the required configuration, run the sample by using the following command:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer/physicalfiletransfer.xml
```

timeout

The timeout sample demonstrates how to use Ant tasks to attempt a file transfer and to cancel the transfer if it takes longer than a specified timeout value. The script initiates a file transfer by using the Managed File Transfer filecopy task. The outcome of this transfer is deferred. The script uses the Managed File Transfer “[fte:awaitoutcome Ant task](#)” on page 822 task to wait a given number of seconds for the transfer to complete. If the transfer does not complete in the given time, the Managed File Transfer “[fte:cancel Ant task](#)” on page 825 task is used to cancel the file transfer.

vsamtransfer

> z/OS

> z/OS The vsamtransfer sample demonstrates how to use Ant tasks to transfer from a VSAM data set to another VSAM data set by using Managed File Transfer. Managed File Transfer currently does not support transferring VSAM data sets. The sample script unloads the VSAM data records to a sequential data set by using the `presrc` [nested element](#) to call the executable file `datasetcopy.sh`. The script uses the Managed File Transfer “[fte:filemove Ant task](#)” on page 830 task to transfer the sequential data set from the source agent to the destination agent. The script then uses the `postdst` [nested element](#) to call the `loadvsam.jcl` script. This JCL script loads the transferred data set records into a destination VSAM data set. This sample uses JCL for the destination call to demonstrate this language option. The same result can also be achieved by using a second shell script instead.

> z/OS

This sample does not require the source and destination data sets to be VSAM. The sample works for any data sets if the source and destination data sets are of the same type.

> z/OS

For this sample to work correctly, you must put the `datasetcopy.sh` script on the command path of the source agent and the `loadvsam.jcl` script on the command path of the destination agent. For more information about setting the command path of an agent, see [commandPath](#).

zip

The zip sample is made up of two scripts: `zip.xml` and `zipfiles.xml`. The sample demonstrates how to use the `presrc` [nested element](#) inside the Managed File Transfer “[fte:filemove Ant task](#)” on page 830 task to run an Ant script before performing a file transfer move operation. The `zipfiles.xml` script called by the `presrc` nested element in the `zip.xml` script compresses the contents of a directory. The `zip.xml` script transfers the compressed file. This sample requires that the `zipfiles.xml` Ant script is present on the command path of the source agent. This is because the `zipfiles.xml` Ant script contains the target used to compress the contents of the directory at the source agent. For more information about setting the command path of an agent, see [commandPath](#).

Customizing MFT with user exits

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Important: Any code within a user exit is not supported by IBM, and any issues with that code need to be initially investigated by either your enterprise, or the vendor who provided the exit.

Managed File Transfer provides points in the code where Managed File Transfer can pass control to a program that you have written (a user exit routine). These points are known as user exit points. Managed File Transfer can then resume control when your program has finished its work. You do not have to use any of the user exits, but they are useful if you want to extend and customize the function of your Managed File Transfer system to meet your specific requirements.

There are two points during file transfer processing where you can invoke a user exit at the source system and two points during file transfer processing where you can invoke a user exit at the destination system. The following table summarizes each of these user exit points and the Java interface that you must implement to use the exit points.

<i>Table 23. Summary of source-side and destination-side exit points and Java interfaces</i>	
Exit point	Java interface to implement
Source-side exit points:	
Before the entire file transfer starts	SourceTransferStartExit.java
After the entire file transfer is complete	SourceTransferEndExit.java
Destination-side exit points:	
Before the entire file transfer starts	DestinationTransferStartExit.java
After the entire file transfer is complete	DestinationTransferEndExit.java

The user exits are invoked in the following order:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Changes made by the SourceTransferStartExit and DestinationTransferStartExit exits are propagated as input to subsequent exits. For example if the SourceTransferStartExit exit modifies the transfer metadata, the changes are reflected in the input transfer metadata to the other exits.

User exits and program calls are invoked in the following order:

- SourceTransferStartExit(onSourceTransferStart).
- PRE_SOURCE Command.
- DestinationTransferStartExits(onDestinationTransferStart).
- PRE_DESTINATION Command.
- The Transfer request is performed.
- DestinationTransferEndExits(onDestinationTransferEnd).
- POST_DESTINATION Command.
- SourceTransferEndExits(onSourceTransferEnd).
- POST_SOURCE Command.

Notes:

1. The **DestinationTransferEndExits** is run only when the transfer completes, either successfully or partially successfully.
2. The **postDestinationCall** is run only when the transfer completes, either successfully or partially successfully.
3. The **SourceTransferEndExits** is run for successful, partially successful, or failed transfers.
4. The **postSourceCall** is called only if:
 - The transfer was not canceled.
 - There is a successful or partially successful outcome.
 - Any post-destination transfer programs ran successfully.

Building your user exit

The interfaces to build a user exit are contained in `MQ_INSTALL_DIRECTORY/mqft/lib/com.ibm.wmqfte.exitroutines.api.jar`. You must include this .jar file in the class path when you build your exit. To run the exit, extract the exit as a .jar file and place this .jar file in a directory as described in the following section.

User exit locations

You can store your user exit routines in two possible locations:

- The `exits` directory. There is an `exits` directory under each agent directory. For example:
`var\mqm\mqft\config\QM_JUPITER\agents\AGENT1\exits`
- You can set the `exitClassPath` property to specify an alternative location. If there are exit classes in both the `exits` directory and the class path set by `exitClassPath`, the classes in the `exits` directory take priority, which means that if there are classes in both locations with the same name, the classes in the `exits` directory take priority.

Configuring an agent to use user exits

There are four agent properties that can be set to specify the user exits that an agent invokes. These agent properties are `sourceTransferStartExitClasses`, `sourceTransferEndExitClasses`, `destinationTransferStartExitClasses`, and `destinationTransferEndExitClasses`. For information about how to use these properties, see [“MFT Agent properties for user exits” on page 853](#).

Running user exits on protocol bridge agents

When the source agent invokes the exit, it passes the exit a list of the source items for the transfer. For normal agents, this is a list of fully-qualified filenames. Because the files should be local (or accessible via a mount), then the exit is able to access it and encrypt it.

However, for a Protocol Bridge Agent, the entries in the list are of the following format:

```
"<file server identifier>:<fully-qualified file name of the file on the remote file server>"
```

For each entry in the list, the exit needs to connect to the file server first (using either the FTP, FTPS or SFTP protocols), download the file, encrypt it locally and then upload the encrypted file back to the file server.

Running user exits on Connect:Direct bridge agents

You cannot run user exits on Connect:Direct bridge agents.

MFT source and destination user exits

Directory separators

Directory separators in source file specifications are always represented using forward slash (/) characters, regardless of how you have specified directory separators in the **fteCreateTransfer** command or in the IBM MQ Explorer. You must take this into account when you write an exit. For example, if you want to check that the following source file exists: `c:\a\b.txt` and you have specified this source file using the **fteCreateTransfer** command or the IBM MQ Explorer, note the file name is actually stored as: `c:/a/b.txt` So if you search for the original string of `c:\a\b.txt`, you will not find a match.

Source side exit points

Before the entire file transfer starts

This exit is called by the source agent when a transfer request is next in the list of pending transfers and the transfer is about to start.

Example uses of this exit point are to send files in stages to a directory that the agent has read/write access to using an external command, or to rename the files on the destination system.

Pass the following arguments to this exit:

- Source agent name
- Destination agent name

- Environment metadata
- Transfer metadata
- File specifications (including file metadata)

The data returned from this exit is as follows:

- Updated transfer metadata. Entries can be added, modified, and deleted.
- Updated list of file specifications, which consists of source file name and destination file name pairs. Entries can be added, modified, and deleted
- Indicator that specifies whether to continue the transfer
- String to insert to the Transfer Log.

Implement the [SourceTransferStartExit.java](#) interface to call user exit code at this exit point.

After the entire file transfer is complete

This exit is called by the source agent after the entire file transfer has completed.

An example use of this exit point is to perform some completion tasks, such as sending an e-mail or an IBM MQ message to flag that the transfer has completed.

Pass the following arguments to this exit:

- Transfer exit result
- Source agent name
- Destination agent name
- Environment metadata
- Transfer metadata
- File results

The data returned from this exit is as follows:

- Updated string to insert to the Transfer Log.

Implement the [SourceTransferEndExit.java](#) interface to call user exit code at this exit point.

Destination side exit points

Before the entire file transfer starts

An example use of this exit point is to validate the permissions at the destination.

Pass the following arguments to this exit:

- Source agent name
- Destination agent name
- Environment metadata
- Transfer metadata
- File specifications

The data returned from this exit is as follows:

- Updated set of destination file names. Entries can be modified but not added or deleted.
- Indicator that specifies whether to continue the transfer
- String to insert into the Transfer Log.

Implement the [DestinationTransferStartExit.java](#) interface to call user exit code at this exit point.

After the entire file transfer is complete

An example use of this user exit is to start a batch process that uses the transferred files or to send an e-mail if the transfer has failed.

Pass the following arguments to this exit:

- Transfer exit result
- Source agent name
- Destination agent name
- Environment metadata
- Transfer metadata
- File results

The data returned from this exit is as follows:

- Updated string to insert to the Transfer Log.

Implement the [DestinationTransferEndExit.java](#) interface to call user exit code at this exit point.

Related concepts

[“Java interfaces for MFT user exits” on page 855](#)

Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference

[“Enabling remote debugging for MFT user exits” on page 252](#)

While you are developing your user exits, you might want to use a debugger to help locate problems in your code.

[“Sample MFT source transfer user exit” on page 252](#)

[“MFT resource monitor user exits” on page 849](#)

Resource monitor user exits allow you to configure custom code to run when a monitor's trigger condition is satisfied, before the associated task is started.

Using MFT transfer I/O user exits

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

Usually for MFT transfers, an agent selects from one of the built-in I/O providers to interact with the appropriate file systems for the transfer. Built-in I/O providers support the following types of file system:

- Regular UNIX-type and Windows-type file systems
-  z/OS sequential and partitioned data sets (on z/OS only)
-  IBM i native save files (on IBM i only)
- IBM MQ queues
- Remote FTP and SFTP protocol servers (for protocol bridge agents only)
- Remote Connect:Direct nodes (for Connect:Direct bridge agents only)

For file systems that are not supported, or where you require custom I/O behavior, you can write a transfer I/O user exit.

Transfer I/O user exits use the existing infrastructure for user exits. However, these transfer I/O user exits differ from other user exits because their function is accessed multiple times throughout the transfer for each file.

Use the agent property `IOExitClasses` (in the agent `.properties` file) to specify which I/O exit classes to load. Separate each exit class with a comma, for example:

```
IOExitClasses=testExits.TestExit1,testExits.testExit2
```

The Java interfaces for the transfer I/O user exits are as follows:

IOExit

The main entry point used to determine if the I/O exit is used. This instance is responsible for making IOExitPath instances.

You need specify only the IOExit I/O exit interface for the agent property IOExitClasses.

IOExitPath

Represents an abstract interface; for example, a data container or wildcard representing a set of data containers. You cannot create a class instance that implements this interface. The interface allows the path to be examined and derived paths to be listed. The IOExitResourcePath and IOExitWildcardPath interfaces extend IOExitPath.

IOExitChannel

Enables data to be read from or written to an IOExitPath resource.

IOExitRecordChannel

Extends the IOExitChannel interface for record-oriented IOExitPath resources, which enables data to be read from or written to an IOExitPath resource in multiples of records.

IOExitLock

Represents a lock on an IOExitPath resource for shared or exclusive access.

IOExitRecordResourcePath

Extends the IOExitResourcePath interface to represent a data container for a record-oriented file; for example, a z/OS data set. You can use the interface to locate data and to create IOExitRecordChannel instances for read or write operations.

IOExitResourcePath

Extends the IOExitPath interface to represent a data container; for example, a file or directory. You can use the interface to locate data. If the interface represents a directory, you can use the listPaths method to return a list of paths.

IOExitWildcardPath

Extends the IOExitPath interface to represent a path that denotes a wildcard. You can use this interface to match multiple IOExitResourcePaths.

IOExitProperties

Specifies properties that determine how Managed File Transfer handles IOExitPath for certain aspects of I/O. For example, whether to use intermediate files or whether to reread a resource from the beginning if a transfer is restarted.

Sample MFT on IBM i user exits

Managed File Transfer provides sample user exits specific to IBM i with your installation. The samples are in the directories *MQMFT_install_dir/samples/ioexit-IBMi* and *MQMFT_install_dir/samples/userexit-IBMi*.

com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit

The com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit sample user exit transfers files in the QDLS file system on IBM i. After the exit is installed, any transfers to files that begin with /QDLS automatically use the exit.

To install this exit, complete the following steps:

1. Copy the com.ibm.wmqfte.samples.ibm.i.ioexits.jar file from the *WMQFTE_install_dir/samples/ioexit-IBMi* directory to the agent's exits directory.
2. Add com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit to the IOExitClasses property.

3. Restart the agent.

com.ibm.wmqfte.exit.user.ibm1.FileMemberMonitorExit

The `com.ibm.wmqfte.exit.user.ibm1.FileMemberMonitorExit` sample user exit behaves like an MFT file monitor and automatically transfers physical file members from an IBM i library.

To run this exit, specify a value for the "library.qsys.monitor" metadata field (using the `-md` parameter, for example). This parameter takes an IFS-style path to a file member and can contain file and member wildcards. For example, `/QSYS.LIB/FOO.LIB/BAR.FILE/*.MBR`, `/QSYS.LIB/FOO.LIB/*.FILE/BAR.MBR`, `/QSYS.LIB/FOO.LIB/*.FILE/*.MBR`.

This sample exit also has an optional metadata field "naming.scheme.qsys.monitor", which you can use to determine the naming scheme that is used during the transfer. By default, this field is set to "unix," which causes the destination file to be called `FOO.MBR`. You can also specify the value "ibmi" to use the IBM i FTP FILE.MEMBER scheme, for example, `/QSYS.LIB/FOO.LIB/BAR.FILE/BAZ.MBR` is transferred as `BAR.BAZ`.

To install this exit, complete the following steps:

1. Copy the `com.ibm.wmqfte.samples.ibm1.userexits.jar` file from the `WMQFTE_install_dir/samples/userexit-IBMi` directory to the agent's exits directory.
2. Add `com.ibm.wmqfte.exit.user.ibm1.FileMemberMonitorExit` to the `sourceTransferStartExitClasses` property in the `agent.properties` file.
3. Restart the agent.

com.ibm.wmqfte.exit.user.ibm1.EmptyFileDeleteExit

The `com.ibm.wmqfte.exit.user.ibm1.EmptyFileDeleteExit` sample user exit deletes an empty file object when the source file member is deleted as part of the transfer. Because IBM i file objects can potentially hold many members, file objects are treated like directories by MFT. Therefore, you cannot perform a move operation on a file object using MFT; move operations are supported at the member level only. Consequently, when you perform a move operation on a member, the now empty file is left behind. Use this sample exit if you want to delete these empty files as part of the transfer request.

If you specify "true" for the "empty.file.delete" metadata and transfer an `FTEFileMember`, the sample exit deletes the parent file if the file is empty.

To install this exit, complete the following steps:

1. Copy the `com.ibm.wmqfte.samples.ibm1.userexits.jar` file from `WMQFTE_install_dir/samples/userexit-IBMi` to the agent's exits directory.
2. Add `com.ibm.wmqfte.exit.user.ibm1.EmptyFileDeleteExit` to the `sourceTransferStartExitClasses` property in the `agent.properties` file.
3. Restart the agent.

Related reference

["Using MFT transfer I/O user exits" on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

["MFT Agent properties for user exits" on page 853](#)

In addition to the standard properties in the `agent.properties` file, there are several advanced properties specifically for user exit routines. These properties are not included by default so if you want to use any of them, you must manually edit the `agent.properties` file. If you make a change to `agent.properties` file while that agent is running, stop and restart the agent to pick up the changes.

Enabling remote debugging for MFT user exits

While you are developing your user exits, you might want to use a debugger to help locate problems in your code.

Because exits run inside the Java virtual machine that runs the agent, you cannot use the direct debugging support that is typically included in an integrated development environment. However, you can enable remote debugging of the JVM and then connect a suitable remote debugger.

To enable remote debugging, use the standard JVM parameters **-Xdebug** and **-Xrunjdwp**. These properties are passed to the JVM that runs the agent by the **BFG_JVM_PROPERTIES** environment variable. For example, on UNIX the following commands start the agent and cause the JVM to listen for debugger connections on TCP port 8765.

```
export BFG_JVM_PROPERTIES="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765"
fteStartAgent -F TEST_AGENT
```

The agent does not start until the debugger connects. Use the **set** command on Windows instead of the **export** command.

You can also use other communication methods between the debugger and JVM. For example, the JVM can open the connection to the debugger instead of vice versa, or you can use shared memory instead of TCP. See the [Java Platform Debugger Architecture](#) documentation for further details.

You must use the **-F** (foreground) parameter when you start the agent in remote debug mode.

Using the Eclipse debugger

The following steps apply to the remote debugging capability in the Eclipse development environment. You can also use other remote debuggers that are JPDA-compatible.

1. Click **Run > Open Debug Dialog** (or **Run > Debug Configurations** or **Run > Debug Dialog** depending on your version of Eclipse).
2. Double-click **Remote Java Application** in the list of configuration types to create a debug configuration.
3. Complete the configuration fields and save the debug configuration. If you have already started the agent JVM in debug mode, you can connect to the JVM now.

Sample MFT source transfer user exit

```
/*
 * A Sample Source Transfer End Exit that prints information about a transfer to standard
 * output.
 * If the agent is run in the background the output will be sent to the agent's event log file.
 * If
 * the agent is started in the foreground by specifying the -F parameter on the fteStartAgent
 * command the output will be sent to the console.
 *
 * To run the exit execute the following steps:
 *
 * Compile and build the exit into a jar file. You need the following in the class path:
 * {MQ_INSTALLATION_PATH}\mqft\lib\com.ibm.wmqfte.exitroutines.api.jar
 *
 * Put the jar in your agent's exits directory:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\exits\
 *
 * Update the agent's properties file:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\agent.properties
 * to include the following property:
 * sourceTransferEndExitClasses=[packageName.]SampleEndExit
 *
 * Restart agent to pick up the exit
 *
 * Send the agent a transfer request:
```

```

* For example: fteCreateTransfer -sa myAgent -da YourAgent -df output.txt input.txt
*/

import java.util.List;
import java.util.Map;
import java.util.Iterator;

import com.ibm.wmqfte.exitroutine.api.SourceTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.TransferExitResult;
import com.ibm.wmqfte.exitroutine.api.FileTransferResult;

public class SampleEndExit implements SourceTransferEndExit {

    public String onSourceTransferEnd(TransferExitResult transferExitResult,
        String sourceAgentName,
        String destinationAgentName,
        Map<String, String>environmentMetaData,
        Map<String, String>transferMetaData,
        List<FileTransferResult>fileResults) {

        System.out.println("Environment Meta Data: " + environmentMetaData);
        System.out.println("Transfer Meta Data: " + transferMetaData);

        System.out.println("Source agent: " +
            sourceAgentName);
        System.out.println("Destination agent: " +
            destinationAgentName);

        if (fileResults.isEmpty()) {
            System.out.println("No files in the list");
            return "No files";
        }
        else {

            System.out.println("File list: ");

            final Iterator<FileTransferResult> iterator = fileResults.iterator();

            while (iterator.hasNext()){
                final FileTransferResult thisFileSpec = iterator.next();
                System.out.println("Source file spec: " +
                    thisFileSpec.getSourceFileSpecification() +
                    ", Destination file spec: " +
                    thisFileSpec.getDestinationFileSpecification());
            }
            return "Done";
        }
    }
}

```

Sample protocol bridge credential user exit

For information about how to use this sample user exit, see [“Mapping credentials for a file server by using exit classes”](#) on page 214

```

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.StringTokenizer;

import com.ibm.wmqfte.exitroutine.api.CredentialExitResult;
import com.ibm.wmqfte.exitroutine.api.CredentialExitResultCode;
import com.ibm.wmqfte.exitroutine.api.CredentialPassword;
import com.ibm.wmqfte.exitroutine.api.CredentialUserId;
import com.ibm.wmqfte.exitroutine.api.Credentials;
import com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit;

/**
 * A sample protocol bridge credential exit

```

```

*
* This exit reads a properties file that maps mq user ids to server user ids
* and server passwords. The format of each entry in the properties file is:
*
* mqUserId=serverUserId,serverPassword
*
* The location of the properties file is taken from the protocol bridge agent
* property protocolBridgeCredentialConfiguration.
*
* To install the sample exit compile the class and export to a jar file.
* Place the jar file in the exits subdirectory of the agent data directory
* of the protocol bridge agent on which the exit is to be installed.
* In the agent.properties file of the protocol bridge agent set the
* protocolBridgeCredentialExitClasses to SampleCredentialExit
* Create a properties file that contains the mqUserId to serverUserId and
* serverPassword mappings applicable to the agent. In the agent.properties
* file of the protocol bridge agent set the protocolBridgeCredentialConfiguration
* property to the absolute path name of this properties file.
* To activate the changes stop and restart the protocol bridge agent.
*
* For further information on protocol bridge credential exits refer to
* the WebSphere MQ Managed File Transfer documentation online at:
* https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html
*/
public class SampleCredentialExit implements ProtocolBridgeCredentialExit {

    // The map that holds mq user ID to serverUserId and serverPassword mappings
    final private Map<String,Credentials> credentialsMap = new HashMap<String, Credentials>();

    /* (non-Javadoc)
    * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#initialize(java.util.Map)
    */
    public synchronized boolean initialize(Map<String, String> bridgeProperties) {

        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the mq user ID mapping properties file
        final String propertiesFilePath = bridgeProperties.get("protocolBridgeCredentialConfiguration");

        if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
            // The properties file path has not been specified. Output an error and return false
            System.err.println("Error initializing SampleCredentialExit.");
            System.err.println("The location of the mqUserId mapping properties file has not been
specified in the
protocolBridgeCredentialConfiguration property");
            initialisationResult = false;
        }

        if (initialisationResult) {

            // The Properties object that holds mq user ID to serverUserId and serverPassword
            // mappings from the properties file
            final Properties mappingProperties = new Properties();

            // Open and load the properties from the properties file
            final File propertiesFile = new File (propertiesFilePath);
            FileInputStream inputStream = null;
            try {
                // Create a file input stream to the file
                inputStream = new FileInputStream(propertiesFile);

                // Load the properties from the file
                mappingProperties.load(inputStream);
            }
            catch (FileNotFoundException ex) {
                System.err.println("Error initializing SampleCredentialExit.");
                System.err.println("Unable to find the mqUserId mapping properties file: " +
propertiesFilePath);
                initialisationResult = false;
            }
            catch (IOException ex) {
                System.err.println("Error initializing SampleCredentialExit.");
                System.err.println("Error loading the properties from the mqUserId mapping properties
file: " + propertiesFilePath);
                initialisationResult = false;
            }
            finally {
                // Close the inputStream
                if (inputStream != null) {
                    try {
                        inputStream.close();

```



```

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Properties;

import com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2;
import com.ibm.wmqfte.exitroutine.api.ProtocolServerPropertyConstants;

/**
 * A sample protocol bridge properties exit. This exit reads a properties file
 * that contains properties for protocol servers.
 * <p>
 * The format of each entry in the properties file is:
 * {@literal serverName=type://host:port}
 * Ensure there is a default entry such as
 * {@literal default=type://host:port}
 * otherwise the agent will fail to start with a BFGBR0168 as it must have a
 * default server.
 * <p>
 * The location of the properties file is taken from the protocol bridge agent
 * property {@code protocolBridgePropertiesConfiguration}.
 * <p>
 * The methods {@code getCredentialLocation} returns the location of the associated
 * ProtocolBridgeCredentials.xml, this sample it is defined to be stored in a directory
 * defined by the environment variable CREDENTIALSHOME
 * <p>
 * To install the sample exit:
 * <ol>
 * <li>Compile the class and export to a jar file.
 * <li>Place the jar file in the {@code exits} subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the {@code protocolBridgePropertiesExitClasses} to
 * {@code SamplePropertiesExit2}.
 * <li>Create a properties file that contains the appropriate properties to specify the
 * required servers.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the <code>protocolBridgePropertiesConfiguration</code> property to the
 * absolute path name of this properties file.
 * <li>To activate the changes stop and restart the protocol bridge agent.
 * </ol>
 * <p>
 * For further information on protocol bridge properties exits refer to the
 * WebSphere MQ Managed File Transfer documentation online at:
 * <p>
 * {@link https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html}
 */
public class SamplePropertiesExit2 implements ProtocolBridgePropertiesExit2 {

    /**
     * Helper class to encapsulate protocol server information.
     */
    private static class ServerInformation {
        private final String type;
        private final String host;
        private final int port;

        public ServerInformation(String url) {
            int index = url.indexOf("://");
            if (index == -1) throw new IllegalArgumentException("Invalid server URL: "+url);
            type = url.substring(0, index);

            int portIndex = url.indexOf(":", index+3);
            if (portIndex == -1) {
                host = url.substring(index+3);
                port = -1;
            } else {
                host = url.substring(index+3, portIndex);
                port = Integer.parseInt(url.substring(portIndex+1));
            }
        }

        public String getType() {
            return type;
        }

        public String getHost() {
            return host;
        }
    }
}

```

```

        public int getPort() {
            return port;
        }
    }

    /** A {@code Map} that holds information for each configured protocol server */
    final private Map<String, ServerInformation> servers = new HashMap<String, ServerInformation>();

    /* (non-Javadoc)
     * @see
     com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#getProtocolServerProperties(java.lang.String)
     */
    public Properties getProtocolServerProperties(String protocolServerName) {
        // Attempt to get the protocol server information for the given protocol server name
        // If no name has been supplied then this implies the default.
        final ServerInformation info;
        if (protocolServerName == null || protocolServerName.length() == 0) {
            protocolServerName = "default";
        }
        info = servers.get(protocolServerName);

        // Build the return set of properties from the collected protocol server information, when
available.
        // The properties set here is the minimal set of properties to be a valid set.
        final Properties result;
        if (info != null) {
            result = new Properties();
            result.setProperty(ProtocolServerPropertyConstants.SERVER_NAME, protocolServerName);
            result.setProperty(ProtocolServerPropertyConstants.SERVER_TYPE, info.getType());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_HOST_NAME, info.getHost());
            if (info.getPort() != -1)
result.setProperty(ProtocolServerPropertyConstants.SERVER_PORT_VALUE, ""+info.getPort());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_PLATFORM, "UNIX");
            if (info.getType().toUpperCase().startsWith("FTP")) { // FTP & FTPS
                result.setProperty(ProtocolServerPropertyConstants.SERVER_TIMEZONE, "Europe/London");
                result.setProperty(ProtocolServerPropertyConstants.SERVER_LOCALE, "en-GB");
            }
            result.setProperty(ProtocolServerPropertyConstants.SERVER_FILE_ENCODING, "UTF-8");
        } else {
            System.err.println("Error no default protocol file server entry has been supplied");
            result = null;
        }

        return result;
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#initialize(java.util.Map)
     */
    public boolean initialize(Map<String, String> bridgeProperties) {
        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the properties file
        final String propertiesFilePath = bridgeProperties.get("protocolBridgePropertiesConfiguration");
        if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
            // The protocol server properties file path has not been specified. Output an error and
return false
            System.err.println("Error initializing SamplePropertiesExit.");
            System.err.println("The location of the protocol server properties file has not been
specified in the
protocolBridgePropertiesConfiguration property");
            initialisationResult = false;
        }

        if (initialisationResult) {
            // The Properties object that holds protocol server information
            final Properties mappingProperties = new Properties();

            // Open and load the properties from the properties file
            final File propertiesFile = new File (propertiesFilePath);
            FileInputStream inputStream = null;
            try {
                // Create a file input stream to the file
                inputStream = new FileInputStream(propertiesFile);

                // Load the properties from the file
                mappingProperties.load(inputStream);
            } catch (final FileNotFoundException ex) {
                System.err.println("Error initializing SamplePropertiesExit.");
                System.err.println("Unable to find the protocol server properties file: " +

```

```

propertiesFilePath);
    initialisationResult = false;
} catch (final IOException ex) {
    System.err.println("Error initializing SamplePropertiesExit.");
    System.err.println("Error loading the properties from the protocol server properties
file: " + propertiesFilePath);
    initialisationResult = false;
} finally {
    // Close the inputStream
    if (inputStream != null) {
        try {
            inputStream.close();
        } catch (final IOException ex) {
            System.err.println("Error initializing SamplePropertiesExit.");
            System.err.println("Error closing the protocol server properties file: " +
propertiesFilePath);
        }
    }
}

    }
}

    if (initialisationResult) {
        // Populate the map of protocol servers from the properties
        for (Entry<Object, Object> entry : mappingProperties.entrySet()) {
            final String serverName = (String)entry.getKey();
            final ServerInformation info = new ServerInformation((String)entry.getValue());
            servers.put(serverName, info);
        }
    }
}

    return initialisationResult;
}

/* (non-Javadoc)
 * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#shutdown(java.util.Map)
 */
public void shutdown(Map<String, String> bridgeProperties) {
    // Nothing to do in this method because there are no resources that need to be released
}

/* (non-Javadoc)
 * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2#getCredentialLocation()
 */
public String getCredentialLocation() {
    String envLocationPath;
    if (System.getProperty("os.name").toLowerCase().contains("win")) {
        // Windows style
        envLocationPath = "%CREDENTIALSHOME%\\ProtocolBridgeCredentials.xml";
    }
    else {
        // Unix style
        envLocationPath = "$CREDENTIALSHOME/ProtocolBridgeCredentials.xml";
    }
    return envLocationPath;
}
}
}

```

Controlling MFT by putting messages on the agent command queue

You can write an application that controls Managed File Transfer by putting messages on agent command queues.

You can put a message on the command queue of an agent to request that the agent performs one of the following actions:

- Create a file transfer
- Create a scheduled file transfer
- Cancel a file transfer
- Cancel a scheduled file transfer
- Call a command
- Create a monitor

- Delete a monitor
- Return a ping to indicate that the agent is active

To request that the agent performs one of these actions, the message must be in an XML format that complies with one of the following schema:

FileTransfer.xsd

Messages in this format can be used to create a file transfer or scheduled file transfer, to call a command, or to cancel a file transfer or scheduled file transfer. For more information, see [“File transfer request message format”](#) on page 755.

Monitor.xsd

Messages in this format can be used to create or delete a resource monitor. For more information, see [“MFT monitor request message formats”](#) on page 772.

PingAgent.xsd

Messages in this format can be used to ping an agent to check that it is active. For more information, see [“Ping MFT agent request message format”](#) on page 781.

The agent returns a reply to the request messages. The reply message is put to a reply queue that is defined in the request message. The reply message is in an XML format defined by the following schema:

Reply.xsd

For more information, see [“MFT agent reply message format”](#) on page 782.

Troubleshooting MFT

Use the following reference information to help you to diagnose errors in Managed File Transfer:

MFT general troubleshooting

Use the following reference information to help you to diagnose errors in Managed File Transfer:

Related concepts

[“Hints and tips for using MFT”](#) on page 261

Here are some suggestions to help you to make best use of Managed File Transfer.

[“Guidance for running an MFT agent or logger as a Windows service”](#) on page 310

You can run a Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the information in this topic to diagnose the issue.

Related tasks

[“Running trace on MFT”](#) on page 262

You can use the methods described in this section to trace Managed File Transfer.

[“What to do if you receive an error when updating your MFT database schema on an Oracle database”](#) on page 315

You might receive the following error message when updating your database schema to the latest level by using the `ftelog_tables_oracle_702_703.sql` file: ERROR at line 1: ORA-02289: sequence does not exist. This error occurs because the sequences and triggers used by the tables are not in the same schema as the tables.

Related reference

[“Common MFT problems”](#) on page 286

Common problems that might occur in your Managed File Transfer network.

[“What to do if your MFT agent is not listed by the `ftelistagents` command”](#) on page 289

If your agent is not listed by the `ftelistagents` command or is not displayed in the IBM MQ Explorer, or your file transfers are not displayed in the **Transfer Log** of the IBM MQ Explorer, you can carry out a number of problem determination steps to investigate the cause.

[“What to do if your MFT agent process disappears but no diagnostic information is logged”](#) on page 290

On UNIX platforms, if an agent process has disappeared but the agent log files do not contain any explanation, this might be caused by the way the agent has been started.

[“What to do if you think that your file transfer is stuck” on page 299](#)

On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

[“What to do if your protocol bridge agent reports that a file is not found” on page 300](#)

When the protocol bridge agent reports that the SFTP or FTP server that the protocol bridge connects to returns a File not found error message, this message can mean that one of a number of different error cases has occurred.

[“What to do if destination files created by a transfer started by a queue resource monitor contain the wrong data” on page 302](#)

You can create a resource monitor to monitor a queue and transfer a message or a group of messages on a queue to a file. The file name can be specified by using the MQMD message descriptors on the message or the first message in a group. If a message-to-file transfer fails and the message or group is left on the queue, the next time the monitor is triggered it might result in files being created that contain the wrong data.

[“What to do if messages are building up on your SYSTEM.MANAGED.DURABLE queues or filling your file system” on page 303](#)

If your IBM MQ Explorer plug-in uses a durable subscription on the coordination queue manager, messages can build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume Managed File Transfer network, use the IBM MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

[“Examining messages before publication” on page 304](#)

Because agents can connect to IBM WebSphere MQ 6.0 queue managers, agents do not use the direct publication approach introduced in IBM WebSphere MQ 7.0. Instead, agents send ordinary messages to the coordination queue manager that contain an MQRFH header. The MQRFH header requests that the message's payload is published. These messages are sent to the SYSTEM.FTE queue on the coordination queue manager, and the messages are typically published immediately from that queue. If error conditions stop this publication, you can examine the messages on the queue before publication is attempted to help with diagnosis. You can do this by completing these following steps:

[“Possible errors when transferring IBM i save files” on page 305](#)

If you use Managed File Transfer to transfer the same IBM i save file several times, the transfer might fail.

[“Guidance for setting MQ attributes and MFT properties associated with message size” on page 306](#)

You can change IBM MQ attributes and Managed File Transfer properties to affect the behavior of Managed File Transfer when reading or writing messages of various sizes.

[“MFT logger error handling and rejection” on page 315](#)

The Managed File Transfer logger identifies two types of error: per-message errors and general errors.

[“What to do if the MFT logger is started, but no transfer information is being logged to the database” on page 317](#)

The database tables used by the Managed File Transfer logger require the database to have a page size of 8 KB or larger. If the page size of the database is not large enough, the tables are not created properly and you see the error SQLSTATE=42704.

[“fteDisplayVersion” on page 450](#)

Use the **fteDisplayVersion** command to display the version of Managed File Transfer that you have installed.

[“BFGSS0023E errors and how to avoid them” on page 320](#)

If you uninstall a Fix Pack from an installation in order to move back to a previous version of the product, and an agent associated with the installation was involved with managed transfers at the time the uninstallation took place, then that agent cannot start and will report an BFGSS0023E error. You can avoid this error by completing a number of steps that should prevent BFGSS0023E messages from appearing when the agents are restarted.

[“Return codes for MFT” on page 322](#)

Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

Hints and tips for using MFT

Here are some suggestions to help you to make best use of Managed File Transfer.

- If you change the `agent.properties` file, stop and restart the agent to pick up the changes.
- If you start a file transfer and there is no sign of transfer progress and no errors are reported, check that the source agent is running. If the transfer is shown but does not progress, check that the destination agent is also running. You can check the current state of agents in the agent log or verify that the agent is active with an **ftePingAgent** command.
- When you cancel an individual transfer using the **fteCancelTransfer** command, you can use either the source or destination agent in the **-agentName** parameter. However, when you delete a transfer schedule using the **fteDeleteScheduledTransfer** command, you must use the source agent name in the **-agentName** parameter.
- When you create a file transfer the source and destination file paths, either absolute or relative, are significant only on the source and destination agents. The system and directory that the **fteCreateAgent** command is issued from has no relevance to the file being transferred.
- Your default environment setup might not be able to fully support Managed File Transfer, particularly if you are running multiple concurrent transfers. If an agent has an error indicating it has run out of memory, check and update the following parameters as required:
 - **UNIX** For UNIX platforms: run the command: `ulimit -m 1048576` (or approximately 1 GB). This maximum resident set size is enough to allow a maximum of 25 concurrent transfers (25 concurrent transfers is the default for the maximum number of transfers for an agent).
 - For all platforms: set the **BFG_JVM_PROPERTIES** environment variable as follows:
`BFG_JVM_PROPERTIES="-Xmx1024M"`

If you want to allow numbers of concurrent transfers greater than the maximum default of 25, use larger sizes for **ulimit** and **BFG_JVM_PROPERTIES** than those suggested.

Note: For Connect:Direct bridge agents the default for the maximum number of concurrent transfers is 5.

- When you use Managed File Transfer to transfer files in text mode between different platforms, the default file encoding of the source platform might not be supported by the destination platform. This causes a transfer to fail with the following error:

```
BFGI00058E: The transfer source encoding xxx is illegal or for an unsupported character set.
```

You can resolve this error by setting the source encoding to one that is supported by the destination platform using an environment variable. Set the **BFG_JVM_PROPERTIES** system environment variable on the source system as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=xxx"`, where `xxx` is an encoding supported by the destination platform. For example, if you are transferring files in text mode from a Solaris platform to a different platform and the source locale is set to "ja", set **BFG_JVM_PROPERTIES** as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=EUC-JP"`. If the source locale is set to "ja_JP.PCK", set **BFG_JVM_PROPERTIES** as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=Shift_JIS"`.

You can also resolve this error for an individual transfer by using the **-sce** parameter when you start a new transfer. For more information, see [“fteCreateTransfer: start a new file transfer”](#) on page 417.

Related reference

[“Java system properties for MFT”](#) on page 580

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

Running trace on MFT

You can use the methods described in this section to trace Managed File Transfer.

About this task

See “Tracing Managed File Transfer resources on Multiplatforms” on page 265 and “Tracing Managed File Transfer for z/OS resources” on page 272 for detailed information on tracing various Managed File Transfer resources.

fteSetAgentTraceLevel

Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

Purpose

Use this command to switch agent trace on and off or to change the level of agent trace that is set. When you use the **fteSetAgentTraceLevel** command, you do not have to shut down and restart an agent to modify the trace level. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%/trace%PID%.txt`, where `%PID%` is the process ID for the agent instance.



Attention:

Multi When using IBM WebSphere MQ 7.5 or later on Multiplatforms, only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

V9.0.2 **z/OS** For z/OS, for Continuous Delivery, from IBM MQ 9.0.2, the **fteSetAgentTraceLevel** command can be run by either:

- The same userid that the agent process is running as.
- Members of the group specified by the agent property **adminGroup**.

V9.0.0.1 This also applies to the Long Term Support release from IBM MQ 9.0.0 Fix Pack 1.

For more information, see the **adminGroup** property in “The MFT agent.properties file” on page 526.

In IBM WebSphere MQ 7.5, and later, the **fteSetAgentTraceLevel** command also writes a trace for the agent process controller. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/pctrace%PID%/pctrace%PID%.txt`, where `%PID%` is the process ID for the agent instance.

You can also use the command to cause the agent process to generate a Javacore. The agent generates a Javacore file in the following directory: `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name`.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.



Attention:

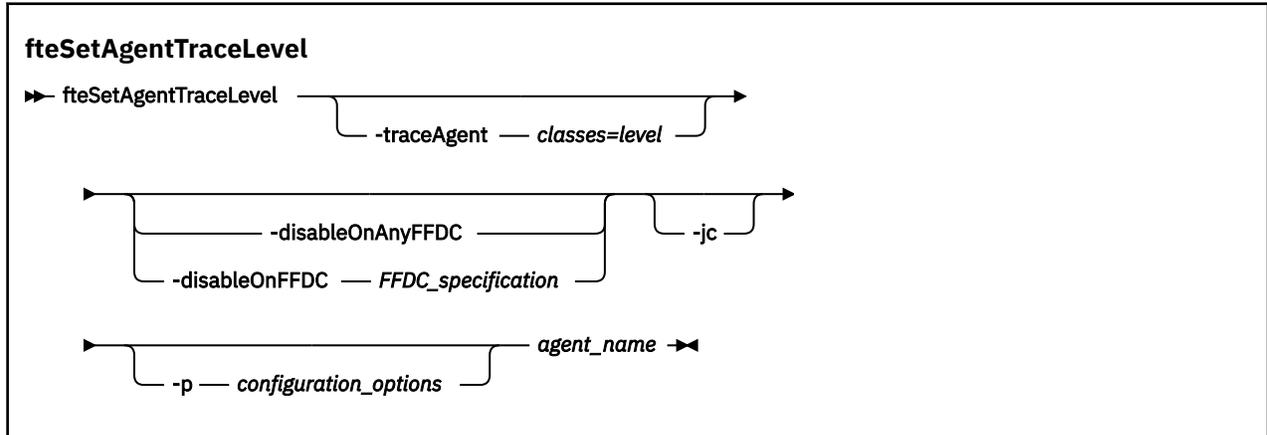
1. You must run this command on the system where the agent is running.
2. The traces and logging do not persist across an agent restart.

If the agent terminates and is restarted by the Process Controller process, the dynamic traces and logs are not in effect until the `agent.properties` file has been updated to include the required trace and log properties.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `agent.properties` file. These properties are described in [Advanced agent properties](#).

Specify the optional `-p` parameter for this command only if you want to use a set of configuration options different from your default set. See [“The MFT agent.properties file”](#) on page 526 for more information.

Syntax



Parameters

-traceAgent classes=level

Required. Level to set the agent trace and which classes to apply the trace to.

V 9.0.5 From IBM MQ 9.0.5, you can specify a colon-separated list of class specifications. This option enables you to set tracing of different classes, and at different levels. For example:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent=all:com.ibm.wmqfte.cmdhandler=moderate  
AGENT1
```

You can still specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes. Use the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

You can substitute *classes* with a Managed File Transfer package name to trace a specific package only. However, because this option captures just a subset of the agent's behavior, you are generally not recommended to use package filtering.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off

Switches the agent trace off but continues to write information to the log files. This is the default option.

flow

Captures data for trace points associated with processing flow in the agent.

moderate

Captures a moderate amount of diagnostic information in the trace.

verbose

Captures a verbose amount of diagnostic information in the trace.

all

Sets agent trace to run on all agent classes.

To start full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =all AGENT_NAME
```

To stop full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =off AGENT_NAME
```

-disableOnAnyFFDC

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The format of the values can be either:

class_name

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc

Optional. Requests that the agent generates a Javacore file. The IBM service team may request that you run the command with this parameter to assist with problem diagnosis. This parameter cannot be used with any other parameter except **-p**.

-p *configuration_options*

Optional. This parameter determines the set of configuration options that is used to set the agent trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name

Required. The name of the Managed File Transfer Agent that you want to set the trace level for.

-? or -h

Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for `AGENT1`:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.agent.Agent` and `com.ibm.wmqfte.cmdhandler` for `AGENT1`:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler=moderate AGENT1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to `off`. All classes starting with `com.ibm.outer` are traced at verbose level except classes starting with `com.ibm.outer.inner`:

```
fteSetAgentTraceLevel -traceAgent com.ibm.outer=verbose AGENT1  
fteSetAgentTraceLevel -traceAgent +com.ibm.outer.inner=off AGENT1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Tracing Managed File Transfer resources on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

About this task

See

- [“Tracing Managed File Transfer agents on Multiplatforms” on page 265](#) for information on how you trace agents.
- [“Tracing Managed File Transfer commands on Multiplatforms” on page 268](#) for information on how you trace commands.

Tracing Managed File Transfer agents on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with an agent, use one of the following options, depending on whether it is possible for you to stop the agent for a short period of time.

If you are unsure which option to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop an agent for a short period of time, collect a trace of the agent from startup.
For more information, see [“Collecting a Managed File Transfer agent trace from startup” on page 266](#).
- If it is not possible for you to stop an agent, then collect a trace dynamically using the **fteSetAgentTraceLevel** command.
For more information, see [“Collecting a Managed File Transfer agent trace dynamically” on page 267](#).

Where it is possible for you to stop an agent for a short period of time, you should collect Managed File Transfer agent trace from startup.

Before you begin

You need to set various properties in the `agent.properties` file for the agent that needs to be traced.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

2. Edit the files and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

3. Stop the agent that needs to be traced, using the **fteStopAgent** command.
4. Start the agent, by running the **fteStartAgent** command.
5. Reproduce the issue.
6. Stop the agent.
7. Edit the `agent.properties` file for the agent, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 266.

This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the agent.

The **fteSetAgentTraceLevel** command allows trace to be collected from a running agent. This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from an agent using the **fteSetAgentTraceLevel** command, complete the following steps.

Procedure

1. Turn on trace for the agent by running the following command:

```
fteSetAgentTraceLevel -traceAgent classes=level agent_name
```

Note: The `-traceAgent` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

An example command might look like this for an agent named AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

2. Reproduce the issue.
3. Turn off trace for the agent by running the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

If an agent is busy, the trace files might wrap quickly and overwrite the information needed to investigate the issue. If this is so, schedule some time to stop the agent then proceed as detailed in the following steps. If you cannot stop the agent for a short period of time, contact your IBM Support representative and discuss alternative trace specifications to use, to reduce the amount of trace data that is being generated.

4. Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

5. Edit the file and add entries for the following properties:

```
traceFiles=number_of_trace_files_to_use  
traceSize=size_of_each_trace_file_in_MB
```

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

```
traceFiles=5  
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

6. Stop the agent, by running the **fteStopAgent** command.

7. Start the agent, by running the **fteStartAgent** command.
8. Enable trace for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent classes=level agent_name
```

Unless otherwise specified by your IBM Support representative, set the **-traceAgent** property to the value `com.ibm.wmqfte=all`.

9. Reproduce the issue.
10. Turn trace off on the agent by running the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

Results

The resultant trace files are written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the agent.

Multi Tracing Managed File Transfer commands on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. You can use this facility to trace commands.

About this task



Attention: Tracing a command only collects information about the processing done by the command. It does not trace any activity that an agent might perform while processing that command.

Procedure

1. Bring up a command prompt, and navigate to the `MQ_INSTALLATION_PATH\bin` directory.
2. Run the command:

Linux UNIX

```
./command_name -trace classes=level -tracePath directory_path command_arguments
```

Windows

```
command_name -trace classes=level -tracePath directory_path command_arguments
```

where

- `command_name` is the name of the command to be traced.
- `classes=level` is the trace level to use, and which classes to enable trace for. Unless otherwise specified by your IBM Support Representative, set this to `com.ibm.wmqfte=all`.
- `directory_path` is the directory where the trace files will be written to.
- `command_arguments` are the arguments that need to be passed to the command, for example, the name of the agent for the **ftePingAgent** command.

Results

The resultant trace files are written to the directory specified by the **-tracePath** parameter.

The trace files are called `trace%PID%.txt.number`, where:

- `%PID%` is the process identifier for the command.

- *number* is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of 0.

However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of 0, the next oldest trace file has a sequence number of 1, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the **-tracePath** parameter, the trace output is written to standard error.

Example

In this example, the **fteListAgents** command is traced, and the trace is written to the C:\trace directory:

```
fteListAgents -trace com.ibm.wmqfte=all -tracePath C:\trace
```

In this example, the **fteCreateTransfer** command is traced, and the trace is written to the /tmp directory:

```
fteCreateTransfer -trace com.ibm.wmqfte=all -tracePath /tmp -t text -sa AGENT1
-da AGENT2 -df /import/transferredfile.txt /export/originalfile.txt
```

The trace file written to /tmp only contains information about the processing performed by the **fteCreateTransfer** command, such as, how the command builds the transfer request message that is sent to the agent, and how long it waits for the agent to send back an acknowledgment indicating that it has received the request. The trace file does not contain any information about the transfer itself.

Multi

Tracing Managed File Transfer standalone loggers on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with a logger, use one of the following options, depending on whether it is possible for you to stop the logger for a short period of time.

If you are unsure which option to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop a logger for a short period of time, collect a trace of the logger from startup.
See [“Collecting a Managed File Transfer standalone logger trace from startup”](#) on page 269.
- If it is not possible for you to stop a logger, then collect a trace dynamically using the **fteSetLoggerTraceLevel** command.
See [“Collecting a Managed File Transfer standalone logger trace dynamically”](#) on page 270.

Multi

Collecting a Managed File Transfer standalone logger trace from startup

Where it is possible for you to stop a logger for a short period of time, you should collect Managed File Transfer logger trace from startup.

Before you begin

You need to set various properties in the `logger.properties` file for the logger that needs to be traced.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

2. Edit the file and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT logger.properties file](#).

3. Stop the logger that needs to be traced, using the **fteStopLogger** command.
4. Start the logger, by running the **fteStartLogger** command.
5. Reproduce the issue.
6. Stop the logger.
7. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 270.

This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the logger.

Multi *Collecting a Managed File Transfer standalone logger trace dynamically*

The **fteSetLoggerTraceLevel** command allows trace to be collected from a running logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a logger using the **fteSetLoggerTraceLevel** command, complete the following steps.

Procedure

1. Turn on trace for the logger by running the following command:

```
fteSetLoggerTraceLevel -traceLogger classes=level logger_name
```

The `-traceLogger` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

2. Reproduce the issue.
3. Turn trace off, using the command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

4. If a logger is busy, then it is possible that the trace files will wrap quickly and overwrite the information needed to investigate the issue.

If you can stop the logger for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

- a) Schedule some time to stop the logger.
- b) Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

- c) Edit the file and add entries for the following properties:

```
traceFiles=number_of_trace_files_to_use  
traceSize=size_of_each_trace_file_in_MB
```

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

```
traceFiles=5  
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT `logger.properties` file](#).

- d) Stop the logger, by running the **`fteStopLogger`** command.
- e) Start the logger, by running the **`fteStartLogger`** command.
- f) Turn trace on for the logger, by running the following command. Unless otherwise specified by your IBM Support representative, set the **`-traceLogger`** property to the value `com.ibm.wmqfte=all`.

```
fteSetLoggerTraceLevel -traceLogger classes=level logger_name
```

- g) Reproduce the issue.
- h) Turn trace off for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

Results

The resultant trace files are written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%` directory, where %PID% is the process identifier for the logger.

Tracing Managed File Transfer for z/OS resources

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

About this task

See

- [“Tracing Managed File Transfer for z/OS agents” on page 272](#) for information on how you trace agents.
- [“Tracing Managed File Transfer for z/OS commands” on page 278](#) for information on how you trace commands.

Tracing Managed File Transfer for z/OS agents

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with an agent, use one of the following options.

If you are unsure which option to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop an agent for a short period of time, collect a trace of the agent from startup.
For more information, see [“Collecting a Managed File Transfer for z/OS agent trace from startup” on page 272](#).
- If it is not possible for you to stop an agent, then collect a trace dynamically using the **fteSetAgentTraceLevel** command.
For more information, see [“Collecting a Managed File Transfer for z/OS agent trace dynamically” on page 275](#).

Collecting a Managed File Transfer for z/OS agent trace from startup

Where it is possible for you to stop an agent for a short period of time, you should collect IBM MQ Managed File Transfer agent trace from startup.

About this task

The way to collect the trace depends on whether the agent is being administered using Unix System Services (USS) or JCL.

If you are unsure which of the following options to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using z/OS UNIX, see [“Collecting an agent trace from startup using USS” on page 273](#).
- If you are using JCL, see [“Collecting an agent trace from startup using JCL” on page 274](#).

To collect a trace of a Managed File Transfer for z/OS agent that is being administered using Unix System Services (USS) from startup, you need to set various properties need in the `agent.properties` file for that agent before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

2. Edit the files and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

3. Stop the agent that needs to be traced, using the **fteStopAgent** command.
4. Start the agent, by running the **fteStartAgent** command.
5. Reproduce the issue.
6. Stop the agent.
7. Edit the `agent.properties` file for the agent, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 273.

This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the agent.

To collect a trace of a Managed File Transfer for z/OS agent that is being administered using JCL from startup, you need to set various properties need in the `agent.properties` file for that agent before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

2. Edit the files and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

3. Locate the data set containing the JCL for the agent that needs to be traced.
4. Submit the BFGAGSP member within the data set to stop the agent.
5. Restart the agent, by submitting the BFGAGST member in the data set .
6. Reproduce the issue.
7. Submit the BFGAGSP member in the data set to stop the agent again.
8. Edit the `agent.properties` file for the agent, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 274.

This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%` directory, where %PID% is the process identifier for the agent.

Collecting a Managed File Transfer for z/OS agent trace dynamically

Where it is not possible for you to stop an agent for a short period of time, you should collect Managed File Transfer for z/OS agent trace dynamically.

About this task

The way to collect the trace depends on whether the agent is being administered using Unix System Services (USS) or JCL.

If you are unsure which of the following options to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
 - Unix System Services (USS), see [“Collecting an agent trace dynamically using USS”](#) on page 275.
 - JCL, see [“Collecting an agent trace dynamically using JCL”](#) on page 276.

Collecting an agent trace dynamically using USS

The **fteSetAgentTraceLevel** command allows trace to be collected from a running agent. This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from a Managed File Transfer for z/OS agent using the **fteSetAgentTraceLevel** command, complete the following steps.

Procedure

1. Turn trace on for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent classes=level agent_name
```

The `-traceAgent` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

2. Reproduce the issue.
3. Turn trace off for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

4. If an agent is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

If you can stop the agent for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

- a) Schedule some time to stop the agent.
- b) Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

- c) Edit the file and add entries for the following properties:

```
traceFiles=number_of_trace_files_to_use  
traceSize=size_of_each_trace_file_in_MB
```

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

```
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

- d) Stop the agent, by running the **fteStopAgent** command.
- e) Start the agent, by running the **fteStartAgent** command.
- f) Turn trace on for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent trace_specification agent_name
```

- g) Reproduce the issue.
- h) Turn trace off for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

Results

The resultant trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the agent.

Collecting an agent trace dynamically using JCL

You can use the BFGAGTC member within the data set containing the JCL, for the agent that needs to be traced, to collect trace from a running Managed File Transfer for z/OS agent. This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from an agent using the BFGAGTC member, complete the following steps.

Procedure

1. Locate the data set containing the JCL for the agent that needs to be traced.
2. Edit the BFGAGTC member within the data set, and locate the line that contains the text:

```
-traceAgent
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by your IBM Support representative, leave this value as is.

3. Submit the BFGAGTC member.

4. Reproduce the issue.
5. Edit the BFGAGTC member again, and set the **-traceAgent** parameter to *=off*, as shown:

```
-traceAgent =off +
```

6. Submit the BFGAGTC member again, to turn trace off.
7. If an agent is busy, then it is possible that the trace files will wrap quickly and overwrite the information needed to investigate the issue.

In this situation there are two options:

a) The first option is to:

- i) Schedule some time to stop the agent.
- ii) Locate the `agent.properties` file for the agent that needs to be traced.
The `agent.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/agents/agent_name` directory.
- iii) Edit the files and add entries for the following properties:

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

```
traceFiles=5  
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see [The MFT agent.properties file](#).

- iv) Locate the data set containing the JCL for the agent that needs to be traced.
- v) Submit the BFGAGSP member within the data set to stop the agent.
- vi) Restart the agent, by submitting the BFGAGST member in the data set.
- vii) Edit the BFGAGTC member within the data set, and locate the line that contains the text:

```
-traceAgent
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by your IBM Support representative, leave this value as is.

- viii) When it is time to enable trace, submit the BFGAGTC member.
- ix) Reproduce the issue.
- x) Edit the BFGAGTC member again, and set the **-traceAgent** parameter to *=off*, as shown:

```
-traceAgent =off +
```

- xi) Submit the BFGAGTC member again, to turn trace off.
- b) The second option is to contact your IBM Support representative, if it is not possible to stop the agent for a short period of time.
You can then discuss alternative trace specifications to use, in order to reduce the amount of trace data that is being generated.

Results

The resultant trace files are written to the *BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%* directory, where %PID% is the process identifier for the agent.

Tracing Managed File Transfer for z/OS commands

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. You can use this facility to trace commands.

About this task



Attention: Tracing a command only collects information about the processing done by the command. It does not trace any activity that an agent might perform while processing that command.

The way to collect the trace depends on whether the command is being run using either Unix System Services (USS) or JCL.

If you are unsure which option to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using z/OS UNIX, see [“Collecting an agent trace from startup using USS”](#) on page 273.
- If you are using JCL, see [“Collecting an agent trace from startup using JCL”](#) on page 274.

Collecting a trace of a command using USS

To collect a trace of a Managed File Transfer for z/OS command using Unix System Services (USS) carry out the following procedure.

Procedure

1. Bring up a command prompt, and navigate to the *BFG_PROD/bin* directory.
2. Run the command:

```
./command_name -trace classes=level -tracePath directory_path command_arguments
```

where

- *command_name* is the name of the command to be traced.
- *classes=level* is the trace level to use, and which classes to enable trace for. Unless otherwise specified by your IBM Support Representative, set this to `com.ibm.wmqfte=all`.
- *directory_path* is the directory where the trace files will be written to.
- *command_arguments* are the arguments that need to be passed to the command, for example, the name of the agent for the **ftePingAgent** command.

Results

The resultant trace files are written to the directory specified by the **-tracePath** parameter.

The trace files are called `trace%PID%.txt.number`, where:

- `%PID%` is the process identifier for the command.
- `number` is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of 0.

However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of 0, the next oldest trace file has a sequence number of 1, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the `-tracePath` parameter, the trace output is written to standard error.

Example

In this example, the `fteListAgents` command is traced, and the trace is written to the `/u/fteuser` directory:

```
./fteListAgents -trace com.ibm.wmqfte=all -tracePath /u/fteuser
```

In this example, the `fteCreateTransfer` command is traced, and the trace is written to the `/tmp` directory:

```
./fteCreateTransfer -trace com.ibm.wmqfte=all -tracePath /tmp -t text -sa AGENT1  
-da AGENT2 -df /tmp/IEEUJV.txt "'SYS1.SAMPLIB(IEEUJV)'"
```

The trace file written to `/tmp` only contains information about the processing performed by the `fteCreateTransfer` command, such as, how the command builds the transfer request message that is sent to the agent, and how long it waits for the agent to send back an acknowledgment indicating that it has received the request. The trace file does not contain any information about the transfer itself.

Collecting a trace of a command using JCL

To collect a trace of a Managed File Transfer for z/OS command that is being submitted using JCL you need to complete the following steps.

Procedure

1. Locate the data set containing the JCL for the command that needs to be traced.
2. Within the data set, locate the member for that command.
3. Edit the member, and locate the line that contains the name of the command that needs to be traced. Modify this line so that it includes the text after the command name and before the `+` sign:

```
-trace classes=level -tracePath directory_path
```

where:

- `classes=level` is the trace level to use, and which classes to enable trace for. Unless otherwise specified by your IBM Support Representative, set this to `com.ibm.wmqfte=all`.
- `directory_path` is the USS directory where the trace files will be written to.

4. Submit the member.
5. After the issue has been reproduced, edit the member again and remove the text:

```
-trace classes=level -tracePath directory_path
```

that you added in Step “3” on page 279.

Results

The resultant trace files are written to the directory specified by the `-tracePath` parameter.

The trace files are called `trace%PID%.txt.number`, where:

- `%PID%` is the process identifier for the command.
- `number` is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of 0.

However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of 0, the next oldest trace file has a sequence number of 1, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the `-tracePath` parameter, the trace output is written to standard error.

Example

In this example, the member BFGMNL1 has been modified to trace the `fteListMonitors` command:

```
/******  
/* * <copyright  
/* * notice="lm-source"  
/* * pids="5655-MF9"  
/* * years="2013,2016"  
/* * crc="3927276320" >  
/* * Licensed Materials - Property of IBM  
/* *  
/* * 5655-MF9  
/* *  
/* * (C) Copyright IBM Corp. 2013, 2024. All Rights Reserved.  
/* * </copyright>  
/******  
/* * fteListMonitors  
/******  
//BFGCMD EXEC PGM=IKJEFT01,REGION=0M  
//SYSEXEC DD DSN=++LIBRARY++,DISP=SHR  
//SYSTSPRT DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//STDERR DD SYSOUT=*  
//SYSTSIN DD *  
%BFGCMD CMD=fteListMonitors -trace com.ibm.wmqfte=all -tracePath /u/trace +  
-v -p QM1  
/*  
//
```

When the member is submitted, the `fteListMonitors` command writes trace to the USS directory `/u/trace`.

Tracing Managed File Transfer for z/OS standalone database loggers

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with a standalone database logger, use one of the following options.

If you are unsure which option to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop a logger for a short period of time, collect a trace of the logger from startup.

For more information, see [“Collecting a Managed File Transfer for z/OS standalone database logger trace from startup”](#) on page 281.

- If it is not possible for you to stop a logger, then collect a trace dynamically using the **fteSetLoggerTraceLevel** command.

For more information, see [“Collecting a Managed File Transfer for z/OS standalone database logger trace dynamically”](#) on page 283.

z/OS *Collecting a Managed File Transfer for z/OS standalone database logger trace from startup*
Where it is possible for you to stop a logger for a short period of time, you should collect IBM MQ Managed File Transfer logger trace from startup.

About this task

The way to collect the trace depends on whether the logger is being administered using UNIX System Services (USS) or JCL.

If you are unsure which of the following options to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
 - UNIX System Services (USS), see [“Collecting a standalone database logger trace from startup using USS”](#) on page 281.
 - JCL, see [“Collecting a standalone database logger trace from startup using JCL”](#) on page 282.

z/OS *Collecting a standalone database logger trace from startup using USS*
To collect a trace of a Managed File Transfer for z/OS logger that is being administered using Unix System Services (USS) from startup, you need to set various properties in the `logger.properties` file for that logger before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

2. Edit the file and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT logger.properties file](#).

3. Stop the logger that needs to be traced, using the **fteStopLogger** command.
4. Start the logger, by running the **fteStartLogger** command.
5. Reproduce the issue.
6. Stop the logger.
7. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 281.

This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the logger.

Collecting a standalone database logger trace from startup using JCL

To collect a trace of a Managed File Transfer for z/OS logger that is being administered using JCL from startup, you need to set various properties in the `logger.properties` file for that logger before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

2. Edit the file and add entries for the following properties:

- **trace**=*trace specification*

The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT logger.properties file](#).

3. Locate the data set containing the JCL for the logger that needs to be traced.
4. Submit the BFGLGSP member within the data set to stop the logger.
5. Restart the logger, by submitting the BFGLGST member in the data set.

6. Reproduce the issue.
7. Submit the BFGLGSP member in the data set to stop the logger again.
8. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step “2” on page 282.
This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%` directory, where %PID% is the process identifier for the logger.

z/OS *Collecting a Managed File Transfer for z/OS standalone database logger trace dynamically*
Where it is not possible for you to stop a logger for a short period of time, you should collect Managed File Transfer for z/OS logger trace dynamically.

About this task

The way to collect the trace depends on whether the logger is being administered using Unix System Services (USS) or JCL.

If you are unsure which of the following options to use, contact your IBM Support representative and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
 - Unix System Services (USS), see [“Collecting a standalone database logger trace dynamically using USS”](#) on page 283.
 - JCL, see [“Collecting a standalone database logger trace dynamically using JCL”](#) on page 284.

z/OS *Collecting a standalone database logger trace dynamically using USS*
Under (USS), you can use the **fteSetLoggerTraceLevel** command to collect trace from a running logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a Managed File Transfer for z/OS logger using the **fteSetLoggerTraceLevel** command, complete the following steps.

Procedure

1. Turn trace on for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger classes=level logger_name
```

The `-traceLogger` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by your IBM Support representative, set this property to the value `com.ibm.wmqfte=all`.

2. Reproduce the issue.
3. Turn trace off for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

4. If a logger is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

If you can stop the logger for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

- a) Schedule some time to stop the logger.
- b) Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.

- c) Edit the file and add entries for the following properties:

```
traceFiles=number_of_trace_files_to_use  
traceSize=size_of_each_trace_file_in_MB
```

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

```
traceFiles=5  
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT `logger.properties` file](#).

- d) Stop the logger, by running the **`fteStopLogger`** command.
- e) Start the logger, by running the **`fteStartLogger`** command.
- f) Turn trace on for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger trace_specification logger_name
```

- g) Reproduce the issue.
- h) Turn trace off for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

Results

The trace files are written to the `BFG_DATA/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%` directory, where `%PID%` is the process identifier for the logger.

Collecting a standalone database logger trace dynamically using JCL

You can use the `BFGLGTC` member within the dataset containing the JCL, for the logger that needs to be traced, to collect trace from a running Managed File Transfer for z/OS logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a logger using the `BFGLGTC` member, complete the following steps.

Procedure

1. Locate the dataset containing the JCL for the logger that needs to be traced.
2. Edit the BFGLGTC member within the dataset, and locate the line that contains the text:

```
-traceLogger
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by your IBM Support representative, leave this value as is.

3. Submit the BFGLGTC member.
4. Reproduce the issue.
5. Edit the BFGLGTC member again, and set the **-traceLogger** parameter to *=off*, as shown:

```
-traceLogger =off +
```

6. Submit the BFGLGTC member again, to turn trace off.
7. If a logger is busy, then it is possible that the trace files will wrap quickly and overwrite the information needed to investigate the issue.

In this situation there are two options:

a) The first option is to:

- i) Schedule some time to stop the logger.
- ii) Locate the `logger.properties` file for the logger that needs to be traced. The `logger.properties` file can be found in the `BFG_DATA/mqft/config/coordination_qmgr_name/loggers/logger_name` directory.
- iii) Edit the file and add entries for the following properties:

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the logger writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

```
traceFiles=5  
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see [The MFT logger.properties file](#).

- iv) Locate the data set containing the JCL for the logger that needs to be traced.
- v) Submit the BFGLGSP member within the data set to stop the logger.
- vi) Restart the logger, by submitting the BFGLGST member in the data set.
- vii) Edit the BFGLGTC member within the data set, and locate the line that contains the text:

```
-traceLogger
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by your IBM Support representative, leave this value as is.

viii) When it is time to enable trace, submit the BFGLGTC member.

ix) Reproduce the issue.

x) Edit the BFGLGTC member again, and set the **-traceLogger** parameter to *=off*, as shown:

```
-traceLogger =off +
```

xi) Submit the BFGLGTC member again, to turn trace off.

b) The second option is to contact your IBM Support representative, if it is not possible to stop the logger for a short period of time.

You can then discuss alternative trace specifications to use, in order to reduce the amount of trace data that is being generated.

Results

The resultant trace files are written to the *BFG_DATA/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%* directory, where %PID% is the process identifier for the logger.

Common MFT problems

Common problems that might occur in your Managed File Transfer network.

- If a text transfer fails with the following error:

```
BFGI00060E: Text data conversion has failed
```

This can occur for one of two reasons:

1. One or more characters in the source file cannot be converted from the source file code page to the destination file code page. This problem can occur when code pages have different character sets and certain characters cannot be converted between them.

If it is acceptable for conversion of some characters to not be converted, a replacement character sequence can be defined at the destination agent so that the transfer does not fail. Specify the agent property **textReplacementCharacterSequence** to define a replacement character sequence. For more information, see [Table 45 on page 528](#).

2. The source file encoding does not match the default encoding of the source agent. In this case performing a text transfer by using the default settings corrupts the character data. To transfer a source file that does not have the same encoding as the source agent, perform one of the following steps:
 - a. Specify the file encoding in a transfer definition file. For more information, see [“Using transfer definition files” on page 152](#).
 - b. Specify the file encoding by using the **-sce** parameter with the **fteCreateTransfer** command. For more information, see the topic [“fteCreateTransfer: start a new file transfer” on page 417](#).
 - c. Specify the file encoding as part of an Ant move or copy task. For more information, see [“Using Apache Ant with MFT” on page 241](#).

To check that you have selected the correct source file encoding for a transfer perform the following steps:

1. Set the destination file encoding to UTF-8.
2. Transfer the file in text mode.

3. Use a UTF-8 file viewer to view the contents of the file. If all characters in the file are correctly displayed, the source file encoding is correct.

- If you see the following output from the **fteCreateAgent** command:

```
BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a
connection attempt.
The IBM MQ reason code was 2058. The agent's presence will not be published.
```

it indicates that the coordination queue manager cannot be contacted and provides the IBM MQ reason code for why. This information message can indicate that the coordination queue manager is currently unavailable or that you have defined the configuration incorrectly.

- If you are using user exit routines and there is a failure while the user exit is being called or just after the exit has been called, for example a product failure or power cut, it is possible the user exit will be called more than once.
- If you have an agent with a queue manager on a system with an IP address that is assigned by DHCP (rather than a static IP address), *and* the agent connects to that system by using a client TCP/IP connection, you must start the agent with the following system environment variable set:

– **Windows** On Windows:

```
set BFG_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=value"
```

– **UNIX** On UNIX:

```
export BFG_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=value"
```

where *value* is the time interval in seconds between each flush of the cached DNS values of the JVM. If the IP address of the queue manager system is reassigned for any reason (for example, because of a network outage, an IP lease expiry, or a system reboot), the agent reports its lost connection to the queue manager. After the JVM DNS cache is flushed, the agent can successfully reconnect. If this environment variable is not set, the agent cannot reconnect in this scenario without a JVM restart. This behavior is because the JVM internally caches the IP addresses of host names and does not refresh them by default.

- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with Managed File Transfer:

```
BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:
The native JNI library 'mqjbd' was not found. [3=mqjbd]
```

If the LD_LIBRARY_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java (as is currently the case for most platforms), this error occurs.

To resolve this issue, set the Managed File Transfer agent property `javaLibraryPath` to reference the correct location for the library. For example, for `mqjbd` on AIX, set to: `/usr/mqm/java/lib`. For `mqjbd` on Linux, set to: `/opt/mqm/java/lib`

- If you have enabled user authority checking by specifying `authorityChecking=true` in the agent property file and all authority checks are failing even if the user has the required authority on the relevant authority queue:
 - Ensure that the user that runs the agent has ALT_USER access control on the agent queue manager.
- If you have enabled user authority checking by specifying `authorityChecking=true` in the agent property file and IBM MQ error messages are written to the agent `output0.log` file perform one of the following actions:
 - Ignore the messages, the agent is not affected.
 - Grant the user who runs the agent GET authority on the SYSTEM.FTE.AUTH* queues belonging to the agent.
- If you have edited the agent property file and the agent has not picked them up:
 - Restart the agent, to ensure that the agent reads the new properties.

z/OS



- If you are using the agent on z/OS to transfer to a PDS or PDSE data set and an abend occurs, your system might have limited disk space. The abend is likely to have a system completion code of B14 with a return code of 0C, indicating there is no space left.

If you are transferring to a sequential data set, the transfer fails and indicates the out-of-space condition, but the agent remains operational.

- If you are using the agent on z/OS, and the WMQFTEP task generates some Java core dumps before becoming unresponsive, apply OMVS system services APAR OA43472.
- If you see the following output when running a configuration or administration script on z/OS:

```
FSUM7332 syntax error: got (, expecting Newline
```

this output indicates that the environment variable `_BPXK_AUTOCVT=ON` has not been set in the environment where the configuration or administration script is being run. For more information about this environment variable and how to set it, see [“Environment variables for MFT on z/OS”](#) on page 83.

Common MFT problems with JZOS

Here are some suggestions if you encounter problems with JZOS.

- If the JZOS fails to process successfully:
 - Add, `PARM=' +T '` to the JCL. For example:

```
//MQMFT EXEC PGM=JVMLDM86,REGION=0M,PARM=' +T '
```

- Add `set -x` to the environment file

- If you get:

```
JVMJZBL1038E Child shell process exited with exit code: 1
JVMJZBL1042E JZOS batch launcher failed, return code=102
```

This means there was something wrong with your environment file and Managed File Transfer commands. This can be due to invalid paths specified.

- From your environment file, locate the value of **BFG_PROD**.

1. Go into OMVS and use the `ls -ltr` command.

For example, if **BFG_PROD** is `/HMF8800/`, type the command:

```
ls -ltr HMF8800/bin/fteBatch
```

2. Check this file exists, and that the batch job has read permission to the file.

3. Resolve any problems.,

- If the JCL still fails to process correctly:

1. Create a file in USS, for example, `myenv` and use an editor to copy information from the environment file into this `myenv` file.

2. Save this file.

3. From the command line, use the command `chmod +x myenv`, to allow the file to be run.

4. Issue the command `. myenv`. Note, that is (period blank filename).

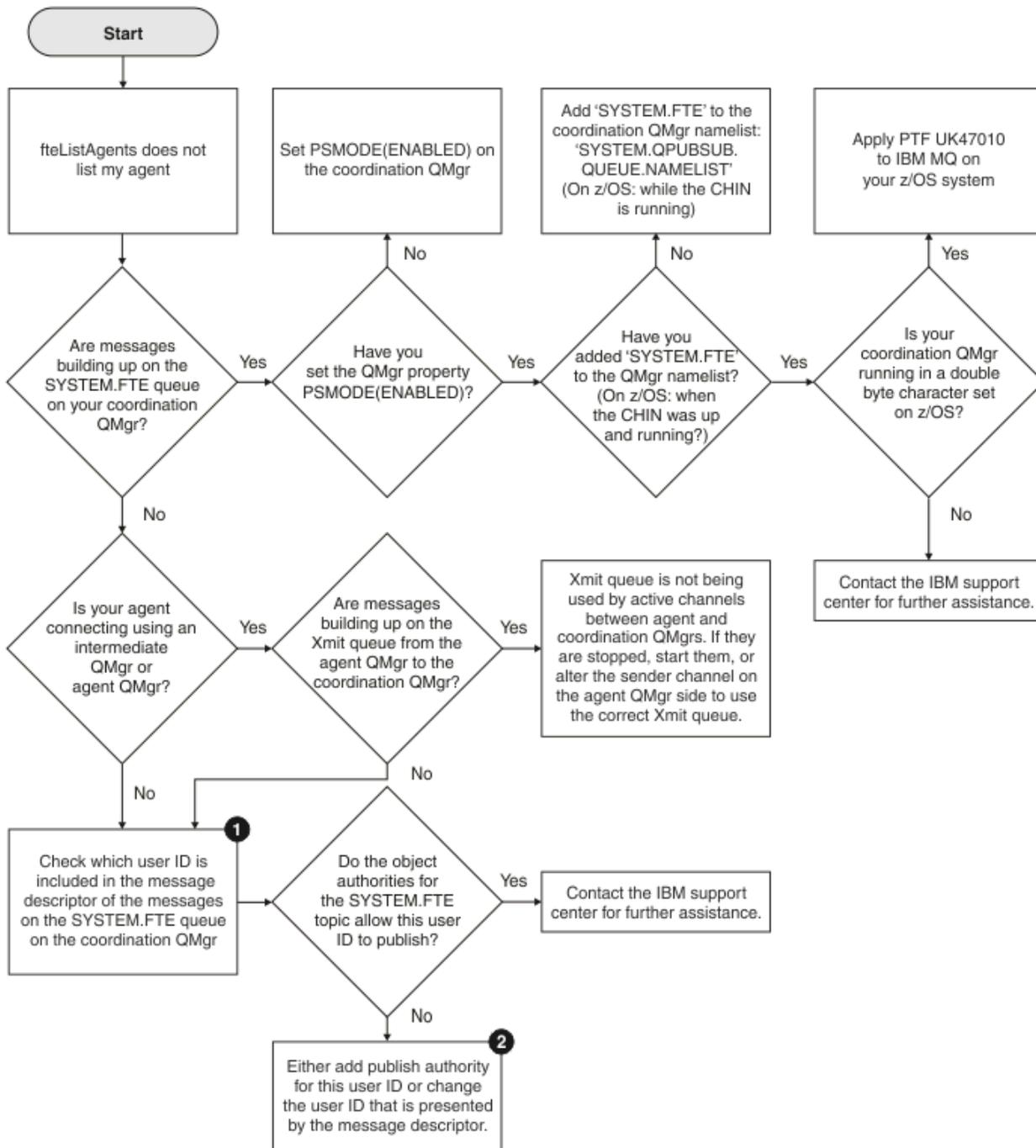
Running this command reports any errors in the `myenv` file.

5. Correct any errors in both the `myenv` and environment files.

What to do if your MFT agent is not listed by the fteListAgents command

If your agent is not listed by the **fteListAgents** command or is not displayed in the IBM MQ Explorer, or your file transfers are not displayed in the **Transfer Log** of the IBM MQ Explorer, you can carry out a number of problem determination steps to investigate the cause.

Use the following flowchart to help you to diagnose problems and decide what action to take next:



Flowchart key:

1. For more information about how to check the user ID that is presented, see [“Examining messages before publication”](#) on page 304. User IDs must conform to the MQ user name 12 character limit. If a user name is longer than 12 characters (Administrator, for example) the user name will be

truncated before being checked for authorisation. In an example using Administrator, the following error message is added to the queue manager error log:

```
AMQ8075: Authorization failed because the SID for entity 'administrato' cannot be obtained.
```

2. For more information about the authority needed for the SYSTEM.FTE queue, see [“Authority to publish MFT Agents log and status messages”](#) on page 346.

What to do if your MFT agent process disappears but no diagnostic information is logged

On UNIX platforms, if an agent process has disappeared but the agent log files do not contain any explanation, this might be caused by the way the agent has been started.

You can check for agent diagnostic information in the following ways:

- Check whether the agent's log files state that the agent has been stopped.
- Check whether the agent lock file `agent.lock` still exists.

If you start the agent from a shell script for example, all child processes associated with that script are removed when the script completes (including the agent process). To keep the agent running past the duration of the script that called the agent, complete the following step:

1. Prefix the **fteStartAgent** command with the **nohup** command to disassociate the **fteStartAgent** process (and any child processes) from the script.

In future when the script terminates, the agent now continues to run.

What to do if an agent is shown as being in an UNKNOWN state

Your agent is running and responds successfully to the **ftePingAgent** command, and items are being transferred normally. However, the **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer Managed File Transfer plugin, report the agent as being in an UNKNOWN state.

Why this problem occurs

Periodically, each agent publishes its status to the SYSTEM.FTE topic on the coordination queue manager. The frequency that an agent publishes its status is controlled by the following agent properties:

agentStatusPublishRateLimit

The maximum rate, in seconds, that the agent republishes its status because of a change in file transfer status. The default value of this property is 30 seconds.

agentStatusPublishRateMin

The minimum rate, in seconds, that the agent publishes its status. This value must be greater than or equal to the value of the **agentStatusPublishRateLimit** property. The default value for the **agentStatusPublishRateMin** property is 300 seconds (or 5 minutes).

The **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer Managed File Transfer (MFT) plugin, use these publications to determine the status of an agent. In order to do this, the commands and the plugin perform the following steps:

1. Connect to the coordination queue manager.
2. Subscribe to the SYSTEM.FTE topic.
3. Receive agent status publications.
4. Create a temporary queue on the coordination queue manager.
5. Put a message to the temporary queue, and save the put time in order to get the current time on the coordination queue manager system.
6. Close the temporary queue.

7. Use the information contained within the publications, and the current time, to determine the status of an agent.
8. Disconnect from the coordination queue manager.

The status message of an agent is considered stale if the difference between the time that it was published, and the current time, is greater than: *The value of the agent property **agentStatusPublishRateMin** (included in the status message) plus the value of the advanced coordination queue manager property **agentStatusJitterTolerance**.*

By default, the **agentStatusJitterTolerance** property has a value of 3000 milliseconds (3 seconds).

If the **agentStatusPublishRateMin** and **agentStatusJitterTolerance** properties are set to their default values, then the status of an agent is considered stale if the difference between the time that it was published, and the current time, is greater than 303 seconds (or 5 minutes 3 seconds).

Any agent with a stale status message is reported by the **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer MFT plugin, as being in an UNKNOWN state.

The status publication of an agent can be stale for one of the following reasons:

1. There is a significant difference in the system time between the system where the agent queue manager is running, and the system where the coordination queue manager is located.
2. The channels between the agent queue manager and the coordination queue manager are stopped (which prevents new status messages from reaching the coordination queue manager).
3. An authorization issue is preventing the agent from publishing its status to the SYSTEM.FTE topic on the coordination queue manager.
4. An agent failure has occurred.

Troubleshooting the problem

There are a number of steps to take to determine why the status of an agent is being reported as UNKNOWN:

1. Check whether the agent is running, by logging on to the agent system. If the agent is stopped, then investigate why it is no longer running. Once it is running again, check whether its status is now being reported correctly.
2. Check that the coordination queue manager is running. If it is not, restart it and then use the **fteListAgents** or **fteShowAgentDetails** command, or the IBM MQ Explorer MFT plugin, to see if the agent status is now being reported correctly.
3. If the agent and coordination queue managers are running, check their error logs to see if there are any authorization issues which are preventing the agent from publishing its status messages. If the logs show that authorization issues are occurring, then ensure that the user running the agent process has the correct authority to publish messages to the SYSTEM.FTE topic on the coordination queue manager.

If the error logs of the queue manager do not report any authorization issues, check the status messages have not got stuck in the IBM MQ network. Verify that all of the sender and receiver channels used to route the messages from the agent queue manager to the coordination queue manager are running.

If the channels are running, then check the transmission queues associated with the channels, to make sure that the status messages are not stuck on them. Also, you should check any dead letter queues for the queue managers to make sure that the status messages have not been placed there for some reason.

4. If the channels are running, and the status messages are flowing through the IBM MQ network, then the next thing to check is that the queue manager's queued publish/subscribe engine is picking up the messages.

The **fteSetupCoordination** command, which is used to define the coordination queue manager, provides you with some MQSC commands that must be run on the coordination queue manager to

configure the queued publish/subscribe engine to receive publications. These commands perform the following steps:

- Create the SYSTEM.FTE topic and its associated topic string.
- Define a local queue called SYSTEM.FTE that will be used to receive incoming status messages.
- Enable the queued publish/subscribe engine, by setting the **PSMODE** attribute on the queue manager to ENABLED.
- Modify the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist, which is used by the queued publish/subscribe engine, so that it includes an entry for the new SYSTEM.FTE queue.

For more information on this, including the MQSC commands that need to be run, see [“fteSetupCoordination”](#) on page 492.

If there are messages on the SYSTEM.FTE queue, then you should check that the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist has been set up correctly and contains an entry for that queue. If the entry is missing, then the queued publish/subscribe engine will not detect any incoming status messages from the agent and will not process them.

You should also ensure that the **PSMODE** attribute on the queue manager is set to ENABLED, which turns on the queued publish/subscribe engine.

5. If the channels are running, and the status messages are flowing through the IBM MQ network and are being picked up from the SYSTEM.FTE queue by the queue manager's queued publish/subscribe engine, then collect the following traces:

- An IBM MQ MFT trace from the agent, covering a time period equal to three times the value of the agent property **agentStatusPublishRateMin**. This ensures that the trace covers the time when the agent is publishing at least three messages containing its status. The trace should be collected dynamically, using the trace specification:

```
com.ibm.wmqfte.statestore.impl.FTEAgentStatusPublisher,  
com.ibm.wmqfte.utils.AgentStatusDetails,  
com.ibm.wmqfte.wmqiface.AgentPublicationUtils,  
com.ibm.wmqfte.wmqiface.RFHMessageFactory=all
```

Note: A reduced amount of trace is output using these strings.

For information on how to enable the trace for agents running on IBM MQ for Multiplatforms, see [“Collecting a Managed File Transfer agent trace dynamically”](#) on page 267.

For information on how to enable the trace for agents running on IBM MQ for z/OS, see [“Collecting a Managed File Transfer for z/OS agent trace dynamically”](#) on page 275.

- A concurrent trace of the queue managers used to route the status messages from the agent queue manager to the coordination queue manager.
- A trace of the **fteListAgents** command, covering the time when the agent is shown as being in an UNKNOWN state. The trace should be collected using the trace specification:

```
com.ibm.wmqfte=all
```

For information on how to enable the trace for commands running on IBM MQ for Multiplatforms, see [“Tracing Managed File Transfer commands on Multiplatforms”](#) on page 268.

For information on how to enable the trace for commands running on IBM MQ for z/OS, see [“Tracing Managed File Transfer for z/OS commands”](#) on page 278.

Once the traces have been collected, they should be made available to IBM Support for analysis.

Related reference

[fteListAgents](#)

[fteShowAgentDetails](#)

MFT agent status values

[The MFT agent.properties file](#)

What to do if **ftePingAgent** times out and reports a **BFGCL0214I** message

ftePingAgent is a useful command-line utility provided with IBM MQ Managed File Transfer that enables you check whether an agent is reachable, and whether it is able to respond to requests.

How the command works

You can use the **ftePingAgent** command to check if an agent is reachable, and whether it is able to process requests. When the command is run, it performs the following steps:

- Connects to the command queue manager for the Managed File Transfer (MFT) topology.
- Creates a temporary reply queue on the command queue manager.

By default, the temporary queue has a name that starts with the prefix WMQFTE. However, you can change this by setting the **dynamicQueuePrefix** property in The MFT command.properties file for the installation.

- Sends a Ping MFT agent request message to the queue `SYSTEM.FTE.COMMAND.agent_name` on the agent queue manager, through the command queue manager. The request message contains the name of the temporary reply queue.
- Waits for a reply message to arrive on the temporary reply queue.

One of the threads within an agent is the `CommandHandler`. This thread gets messages from the `SYSTEM.FTE.COMMAND.agent_name` queue of the agent, and processes them.

If this thread receives a message containing a Ping MFT agent request, it builds an MFT agent reply message, and sends it to the temporary queue on the command queue manager. This message goes through the agent's queue manager.

Once the message arrives on the temporary queue, it is picked up by the **ftePingAgent** command. The command then writes a message similar to the one shown below to the console before exiting:

```
BFGCL0213I: agent <agent_name> responded to ping in 0.088 seconds.
```

The following two diagrams show the flow:

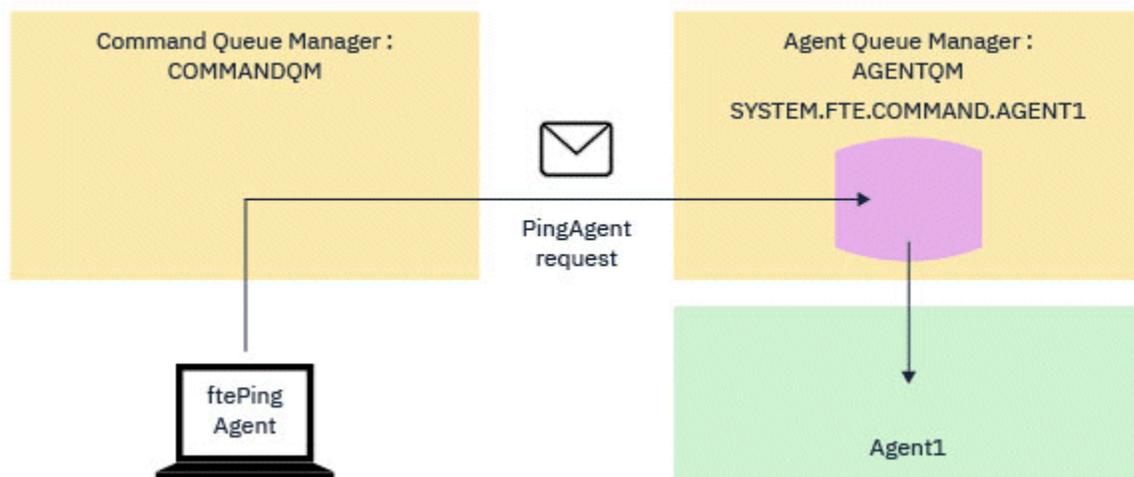


Figure 7. The pingAgent request goes to the `SYSTEM.FTE.COMMAND.agent_name` queue on the agent queue manager, through the command queue manager

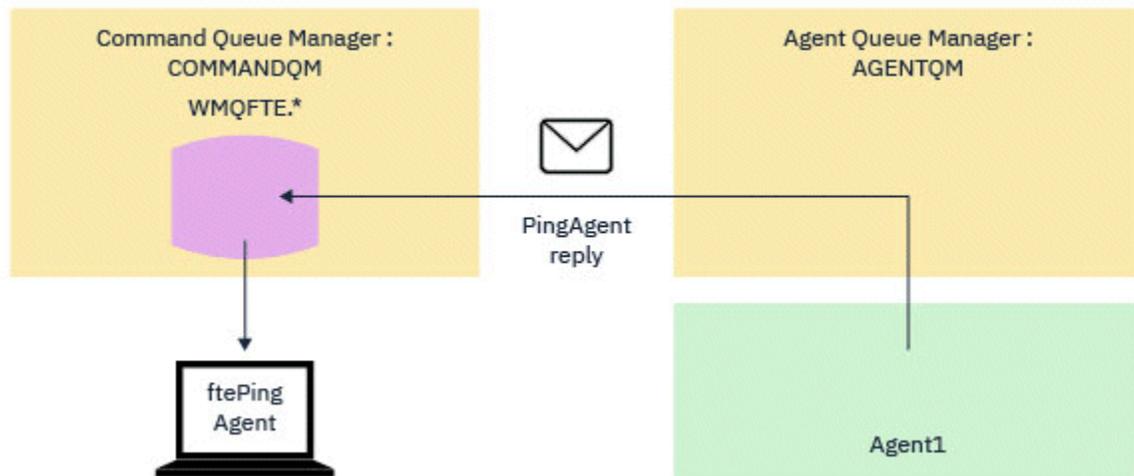


Figure 8. The pingAgent reply comes back through the agent queue manager to the command queue manager.

What to do if the command times out

By default, the **ftePingAgent** command waits for five seconds for the MFT agent reply message to arrive on the temporary queue. If the reply message does not arrive within five seconds, the command writes a BFGCL0214I message to the console. The following message is an example:

```
BFGCL0214I: agent AGENT1 didn't respond to ping after 5 seconds.
```

You can use the following steps to investigate why the reply message did not arrive:

- The first thing to do is check that the agent is running. If it is not, it can not respond to the Ping MFT agent request sent by the command.
- If the agent is running and busy processing requests, it is possible that it might take longer than five seconds to get the Ping MFT agent request and send back the reply.

To see if this is the case you should rerun the **ftePingAgent** command again, using the **-w** parameter to specify a longer wait interval. For example, to specify a 60 second wait interval, issue the following command:

```
ftePingAgent -w 60 AGENT1
```

- If the command still times out, check the path through the IBM MQ network between the command queue manager and the agent queue manager. If one or more channels in the path have failed, the Ping MFT agent request message and/or the MFT agent reply message will be stuck on a transmission queue somewhere. In this situation, you should restart the channels and re-run the **ftePingAgent** command.

If the command still reports a BFGCL0214I message after you have carried out the preceding steps, the Ping MFT agent request and MFT agent reply messages need to be tracked as they flow through the IBM MQ network to see:

- Whether the Ping MFT agent message ever reaches the SYSTEM.FTE.COMMAND.*agent_name* queue.
- If the agent picks up the message up from the queue, and sends back an MFT agent reply message.

To do this, you should carry out the following steps:

- Enable queue manager traces on both the command and agent queue managers.
- Enable trace on the agent dynamically using the trace specification `com.ibm.wmqfte=all`.

The way to do this depends upon the platform the agent is running on. For agents running on:

- IBM MQ for Multiplatforms, see [“Collecting a Managed File Transfer agent trace dynamically”](#) on page 267.

- IBM MQ for z/OS, see [“Collecting a Managed File Transfer for z/OS agent trace dynamically”](#) on page 275.
- Next, run the **ftePingAgent** command with trace enabled, using the trace specification `com.ibm.wmqfte=all`. For information about tracing the command on:
 - IBM MQ for Multiplatforms, see [“Tracing Managed File Transfer commands on Multiplatforms”](#) on page 268.
 - IBM MQ for z/OS, see [“Tracing Managed File Transfer for z/OS commands”](#) on page 278.

When the command times out, stop the agent trace and the queue manager trace. The agent and queue manager traces, along with the trace from the command, should then be made available to IBM support for analysis.

What to do if your MFT agent or logger configuration is not secure

If a Managed File Transfer process detects a condition that a configuration file contains sensitive information, is a keystore or truststore file, and has system-wide read, write, or delete permissions, the process will fail to start if detected at startup time. If the condition was not detected at startup time but was detected at runtime, Managed File Transfer generates a warning message and ignores the contents of the configuration file. This is relevant to the protocol bridge and the Connect:Direct bridge capabilities which reload a configuration if it changes while the agent is running.

Complete the following checks to determine the cause of the problem:

1. Identify the configuration file that has been reported as not secure from the error message provided.
2. Ensure that the file access permissions match the requirements needed. For more information, see [“MFT permissions to access sensitive configuration information”](#) on page 346.
3. Restart the agent or logger. Or, in the case of the protocol bridge or Connect:Direct credentials files, wait for the next reload.

Example

In this example of an error message, a database logger is failing to start:

```
BFGDB0066E: The logger encountered a problem accessing its credentials file and will stop.
Reported error: BFGNV0145E: The 'Everyone' group has access to the file 'C:\mqmftcredentials.xml'.
```

In this example of an error message, a protocol bridge agent is failing to start:

```
BFGI00383E: The security permissions defined for credentials file 'C:\ProtocolBridgeCredentials.xml' do
not meet the
minimum requirements for a file of this type.
Reported problem: BFGNV0145E: The 'Everyone' group has access to the file
C:\ProtocolBridgeCredentials.xml'.
```

Related reference

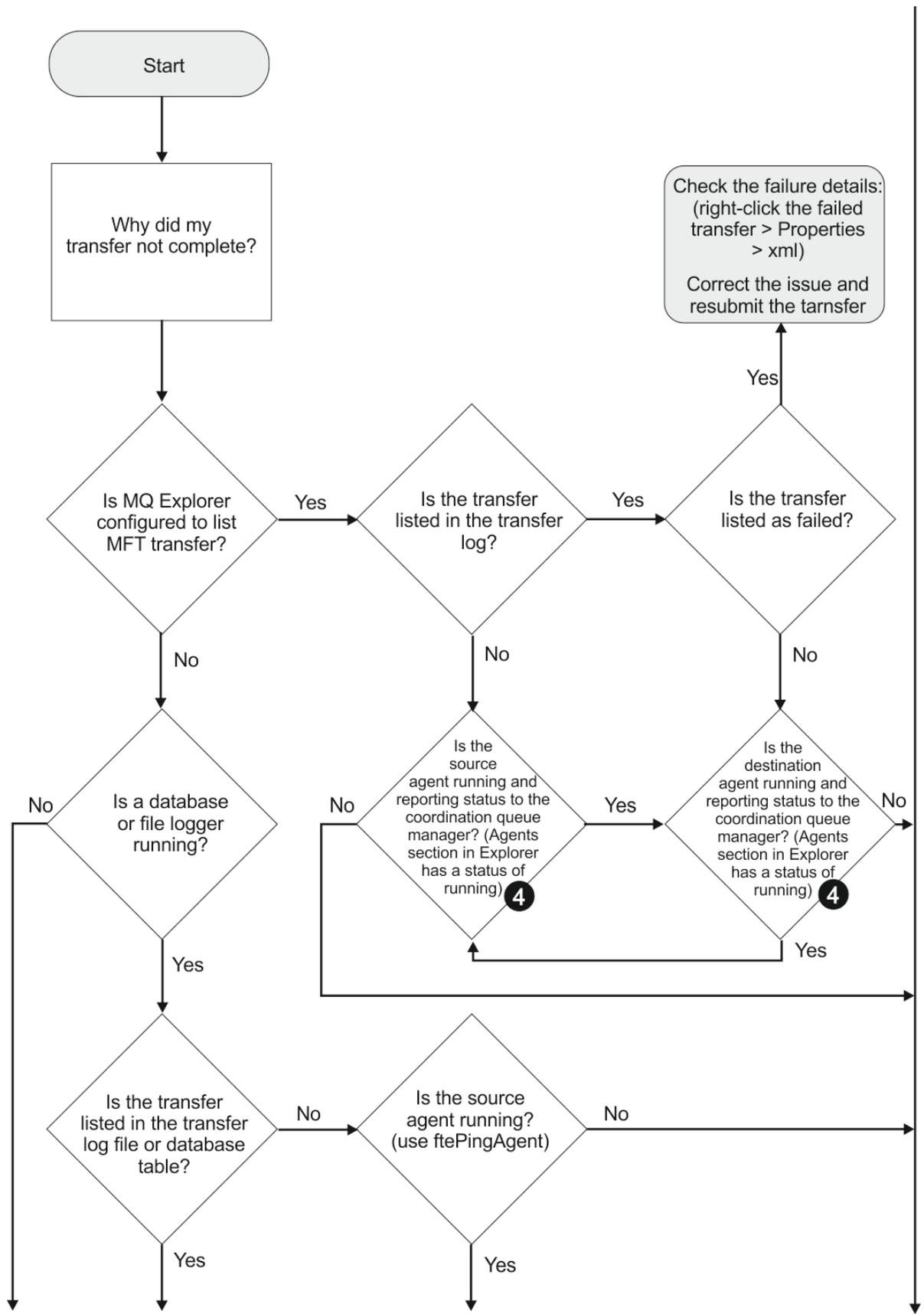
[“MFT permissions to access sensitive configuration information”](#) on page 346

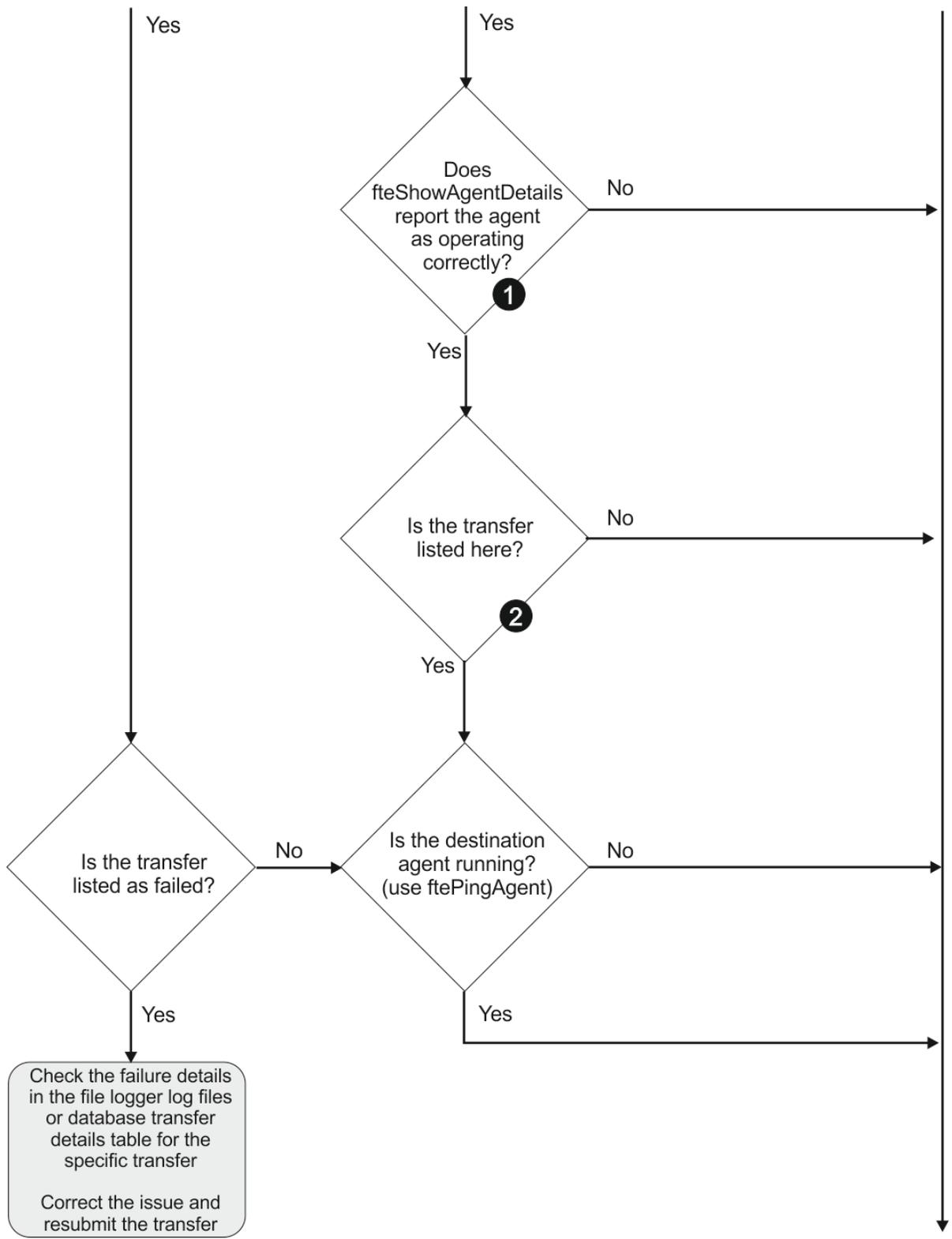
Any file used to store sensitive configuration information, meaning any file referenced from the IBM MQ configuration tree, must not have system-wide read, write, or (where applicable), delete permissions. These restrictions also apply to truststore and keystore files.

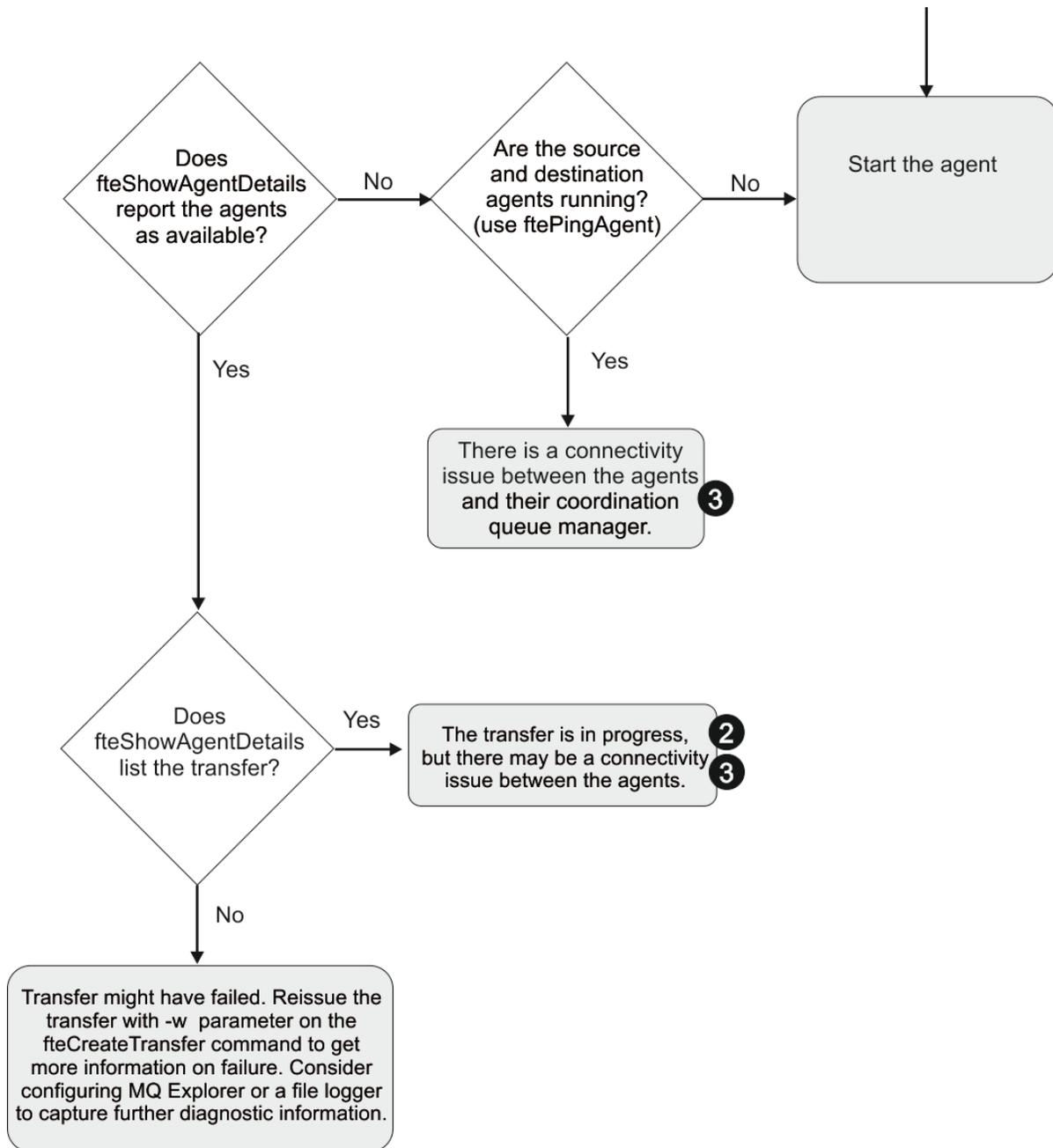
What to do if your transfer does not complete

If your transfer does not complete you can carry out a number of problem determination steps to investigate the cause.

Use the following flowchart to help you to diagnose problems and decide what action to take next:







Flowchart key:

1. Check the agent output`0.log` for errors. If the agent reports it has successfully started, but neither IBM MQ Explorer nor **fteShowAgentDetails** report the agent as running, then check the connectivity between the agent queue manager and the coordination queue manager. It may be that a queue manager to queue manager channel is unavailable.
2. If the source agent lists the transfer ID as an `In progress` transfer but the destination agent does not, there might be a connectivity issue between the source and destination queue managers. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.
3. If both the source and destination agents list the transfer ID as `In progress`, this suggests there has been a connectivity issue between the source and destination queue managers since the transfer was initiated. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the

command.properties file. You can also run this command the other way round, from source to destination.

4. If you have been round this loop already, check whether either of statements are relevant to your situation:
 - Both source and destination agents report as Running, but no transfer is listed. Either the transfer request did not reach the agent command queue, or the agent although reporting as Running, is no longer monitoring the command queue. Check for errors in the source agent output0.log. Use the **ftePingAgent** command from the same machine the transfer was sent from, to the source agent, to verify the connectivity between the command queue manager and the agent queue manager, and that the agent is servicing the command queue.
 - Both source and destination agents report as Running, and the transfer is listed as In progress, recovering. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the command.properties file. You can also run this command the other way round, from source to destination.

What to do if you think that your file transfer is stuck

On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

Complete the following checks to determine the cause of the problem:

1. Use the **ftePingAgent** command, or in the IBM MQ Explorer **Agents** panel right-click on the agent name and select **Ping**, to check whether the source and destination agents are active and responding to new requests. Look at the agent logs to see if there is a current network connection problem.
2. Check whether the destination agent is running at capacity. It might be that there are numerous source agents all requesting file transfers to the same destination agent. Use the **fteShowAgentDetails** command with the **-v** (verbose) parameter, or in the IBM MQ Explorer **Agents** panel right-click on the agent name and select **Properties**, to see the current transfer activity for an agent. If the number of running destination transfers is at or close to the agent's maximum number of destination transfers, that can explain why some transfers for source agents appear to be stuck.
3. Transfers to and from protocol bridge agents enter a recovering state if there is a problem contacting the protocol file server. Look at the agent logs to see if there is a current connection problem.
4. Transfers are processed by an agent in priority order. Therefore in a loaded system, a low-priority transfer can remain in the queued state for some time while the agent is loaded with higher priority transfers. Eventually a low-priority transfer is started if that transfer has been queued for a while, even though there are newer higher priority transfers.

What to do if your scheduled file transfer does not run or is delayed

If you have a scheduled transfer that does not run when it is due or is delayed, it might be because the agent is processing commands on its command queue. Because the agent is busy, scheduled transfers are not checked and are therefore not run.

To work around this problem, use one of the following steps:

- Configure the maxSchedulerRunDelay property in the agent.properties file to set the maximum interval in minutes that the agent waits to check for scheduled transfers. Setting this property ensures that the agent keeps checking for scheduled transfers even when the agent is busy. For more information about the property, see [“The MFT agent.properties file” on page 526](#).
- Alternatively, use a resource monitor instead of a scheduled transfer. Resource monitors work differently from scheduled transfers and are not affected by the agent being busy. For example, if you want an up-to-date file on the destination system, resource monitors reduce network traffic. This is because the file is transferred only when a new version becomes available, rather than the file being

transferred automatically. However, resource monitoring is not supported on protocol bridge agents or Connect:Direct bridge agents.

For more information, see [“Monitoring MFT resources” on page 162.](#)

What to do if your protocol bridge agent reports that a file is not found

When the protocol bridge agent reports that the SFTP or FTP server that the protocol bridge connects to returns a File not found error message, this message can mean that one of a number of different error cases has occurred.

The following possible scenarios can result in a File not found error being returned by the SFTP or FTP server.

- The file does not exist. Check that the file you are attempting to transfer exists on the system hosting the SFTP or FTP server.
- The file path does not exist. Check that the file path exists on the system hosting the SFTP or FTP server. Check that you have entered the file path correctly into the transfer request. If necessary, correct the file path and submit the transfer request again.
- The file is locked by another application. Check whether the file is locked by another application. Wait until the file is no longer locked then submit the transfer request again.
- The file permissions do not allow the file to be read. Check whether the file has the correct file permissions. If necessary, change the file permissions and submit the transfer request again.
- The SFTP or FTP server uses a virtualized root path. If a relative file path is specified in a transfer request, the protocol bridge agent will attempt to convert the relative path into an absolute file path based on the home directory used to login to the protocol server. The Managed File Transfer protocol bridge agent can support only SFTP or FTP servers that allow files to be accessed by their absolute file path. Those protocol servers that allow access to files based only on the current directory are not supported by the protocol bridge agent.

Related concepts

[“The protocol bridge” on page 204](#)

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

What to do if your MFT directory resource monitor is not triggering files

A directory resource monitor polls a directory for files that match a trigger specification. For each file that matches the trigger specification, a transfer request is generated to the agent. When the request is submitted, the triggering file is ignored until the file is changed.

Possible reasons why the files are not triggering

1. The directory resource monitor found a file that matched the trigger specification, but the generated transfer request was invalid and the agent was unable to process the request. The reasons can include the following:
 - Invalid destination agent
 - Missing destination agent
 - Transfer canceled by program invocation

In all these examples, the directory resource monitor marks the triggering file as processed and ignores the file even though the transfer failed.

2. The file is outside the scope of the resource monitor trigger specification. The reasons can include the following:

- Incorrect trigger pattern
- Monitoring the incorrect directory
- Insufficient file permissions
- Failure to connect to remote file system

Why a file can trigger a second transfer

A trigger file can generate a Managed File Transfer transfer request for the following reasons:

- If the presence of the trigger file is detected, when it was not there before.
- If the trigger file has been updated, causing the last modified date to change.

Potential scenarios for a second trigger are:

- The file is removed, then replaced.
- The file is locked by one application, then unlocked by another application.
- The monitor file system fails. For example, if the network connection fails, this can give the appearance of the file being removed, then replaced.
- The file directory is updated by another application, causing the last modified date to change.

Command to set info level output for all resource monitors of an agent

V 9.0.3

In this example, all resource monitors are being monitored because you have not specified a name, or names, of specific resource monitors. The name of the agent is AGENT1.

```
fteSetAgentLogLevel -logMonitor=info AGENT1
```

See [fteSetAgentLogLevel](#) for details of the **logMonitor** parameter, and examples of how you use the different options.

Example of info level output for all resource monitors of an agent

V 9.0.3

```
=====
[21/04/2017 11:08:49:367 IST] BFGUT0036I: Resource monitor event log level has changed to "info" for all resource monitors of
this agent.
=====
```

```
=====
Date      Time          Thread ID   Monitor Name   Event
Description
=====
```

Date	Time	Thread ID	Monitor Name	Event
[21/04/2017 11:08:51:842 IST]		00000023	QMON	Monitor Started
[21/04/2017 11:08:51:844 IST]		00000025	QMON	Start Poll
[21/04/2017 11:08:51:924 IST]		00000023	MON1	Monitor Started
[21/04/2017 11:08:51:925 IST]		00000026	MON1	Start Poll
[21/04/2017 11:08:52:029 IST]		00000026	MON1	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:08:52:055 IST]		00000025	QMON	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:09:51:840 IST]		00000025	QMON	Start Poll
[21/04/2017 11:09:51:875 IST]		00000025	QMON	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:09:51:924 IST]		00000026	MON1	Start Poll
[21/04/2017 11:09:51:969 IST]		00000026	MON1	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:10:51:840 IST]		00000025	QMON	Start Poll
[21/04/2017 11:10:51:924 IST]		00000026	MON1	Start Poll
[21/04/2017 11:10:51:962 IST]		00000025	QMON	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:10:51:963 IST]		00000026	MON1	End Poll
milli seconds. Trigger items matched [0]				
[21/04/2017 11:10:55:063 IST]		00000041	MON1	Monitor Stopped
[21/04/2017 11:10:55:079 IST]		00000041	QMON	Monitor Stopped

Related reference

[“fteSetAgentLogLevel: set MFT protocol bridge agent log level” on page 478](#)

Use the **fteSetAgentLogLevel** command to enable or disable diagnostic information logging for file transfers between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers.

What to do if destination files created by a transfer started by a queue resource monitor contain the wrong data

You can create a resource monitor to monitor a queue and transfer a message or a group of messages on a queue to a file. The file name can be specified by using the MQMD message descriptors on the message or the first message in a group. If a message-to-file transfer fails and the message or group is left on the queue, the next time the monitor is triggered it might result in files being created that contain the wrong data.

Why this problem occurs

1. A message-to-file transfer fails and the message or group is left on the queue.
2. A new message or group arrives on the queue.
3. The new message or group triggers the resource monitor.
4. The resource monitor creates a new transfer that uses the MQMD message descriptors from the new message or group and the data from the first message or group on the queue.
5. Files are created that contain the wrong data.

Avoiding this problem

To avoid experiencing this problem, you must manually create a transfer definition file by using the **fteCreateTransfer** command and edit the <queue> element of the file to include the attribute `groupId="{GROUPID}"`. Then submit the transfer definition file by using the **fteCreateMonitor** command.

Example

In this example: the source agent, which is also the monitoring agent, is called AGENT_MON; the destination agent is called AGENT_DEST; the destination file name is `/out/files/{WMQFTEFileName}`. This example requires that the message has the MQMD message descriptor `WMQFTEFileName` set. The queue being monitored is LIVE_QUEUE.

1. Create a transfer definition file by running the following command:

```
fteCreateTransfer -sa AGENT_MON -da AGENT_DEST -df "/out/files/{WMQFTEFileName}"  
-de error -gt /tmp/TransferDefinition1.xml -sqgi -sq LIVE_QUEUE
```

The transfer definition file `/tmp/TransferDefinition1.xml` is generated.

2. Edit the <queue> element to include the attribute `groupId="{GROUPID}"`. Change the line

```
<queue useGroups="true">LIVE_QUEUE</queue>
```

to

```
<queue useGroups="true" groupId="{GROUPID}">LIVE_QUEUE</queue>
```

This attribute is required so that the transfer reads the group or message that triggered the transfer from the queue instead of the first group or message on the queue.

3. Create the monitor by running the following command:

```
fteCreateMonitor -ma AGENT_MON -mq LIVE_QUEUE -mn QueueMon1 -mt /tmp/TransferDefinition1.xml  
-tr completeGroups -dv WMQFTEFileName=UNKNOWN
```

This monitor polls the queue every 60 seconds to see if a new group or message has arrived on the queue.

What to do if the destination queue is a clustered queue, or an alias to a clustered queue

When using Managed File Transfer to transfer a file into a queue, if you use a destination that is a clustered queue, or an alias to a clustered queue, you get reason code 2085, or 2082. From IBM WebSphere MQ 7.5.0 Fix Pack 4 onwards, this issue is resolved if you set the property `enableClusterQueueInputOutput` to true.

Why this problem occurs

The queue manager name of the destination agent is being appended to the queue name of the `-dq` parameter, when there is no explicit queue manager name on the `-dq`. The reason code 2085, or 2082, occurs because the `queueManager` object cannot be specified on an MQOPEN call when connecting to a clustered MQ `queueManager` that does not have that local clustered queue.

Avoiding this problem

1. Create a clustered queue on the queue manager.
2. Set up a remote queue definition that points to a clustered queue.

Example

This example uses a remote queue definition.

Configuration:

- Source Agent: *SAGENT*
- Source Agent Queue Manager: *SQM*
- Destination Agent: *DAGENT*
- Destination Agent Queue Manager: *DQM*
- The destination queue of the transfer is *CQ6* on queue manager *SQM*

To define remote queue definition *Q6_SQM* on *DQM* to clustered queue *CQ6* in *SQM* (assuming that the clustered queue *CQ6* is already defined in *SQM*), issue the MQSC command on the *DQM* queue manager:

```
define qremote(Q6_SQM) rname(CQ6) rqmname(SQM) xmitq(SQM)
```

Note: `rname` points to the clustered queue.

You can now transfer to the queue. For example:

```
fteCreateTransfer -sa SAGENT -sm SQM -da DAGENT -dm DQM -dq Q6_SQM /tmp/single_record.txt
```

What to do if messages are building up on your SYSTEM.MANAGED.DURABLE queues or filling your file system

If your IBM MQ Explorer plug-in uses a durable subscription on the coordination queue manager, messages can build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume Managed File Transfer network, use the IBM MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

To remove the buildup of messages on the SYSTEM.MANAGED.DURABLE queues, you can perform one of the following actions:

- Start the IBM MQ Explorer that uses the durable subscription. The Managed File Transfer plug-in for IBM MQ Explorer consumes the messages from the queue.
- Delete the messages from the queues manually.

V 9.0.2

You can avoid the build up of messages on durable queues in one of the following ways:

- Specify that the IBM MQ Explorer plug-in uses a non-durable subscription to the coordination queue manager. Perform the following steps in your IBM MQ Explorer:
 1. Select **Window > Preferences > IBM MQ Explorer > Managed File Transfer**
 2. From the **Transfer Log subscription type** list, choose NON_DURABLE.
- Clear durable subscriptions from the coordination queue manager that are created by the IBM MQ Explorer MFT plugin.

The name of the durable subscription is prefixed to show that the subscription was created by the IBM MQ Explorer MFT plugin, the host name, and the name of the user, for example MQExplorer_MFT_Plugin_HOST_TJWatson.

Related reference

“Retaining MFT log messages” on page 99

Managed File Transfer sends file transfer progress and log information to the coordination queue manager. The coordination queue manager publishes this information to any matching subscriptions to the SYSTEM.FTE topic. If there are no subscriptions, this information is not retained.

Examining messages before publication

Because agents can connect to IBM WebSphere MQ 6.0 queue managers, agents do not use the direct publication approach introduced in IBM WebSphere MQ 7.0. Instead, agents send ordinary messages to the coordination queue manager that contain an MQRFH header. The MQRFH header requests that the message's payload is published. These messages are sent to the SYSTEM.FTE queue on the coordination queue manager, and the messages are typically published immediately from that queue. If error conditions stop this publication, you can examine the messages on the queue before publication is attempted to help with diagnosis. You can do this by completing these following steps:

1. Disable the publish/subscribe engine in the coordination queue manager.

You can either complete this step using the IBM MQ Explorer or using MQSC commands. Be aware that this temporarily stops all publish/subscribe activity on the queue manager, including activity unrelated to Managed File Transfer if your coordination queue manager is also used for other purposes.

IBM MQ Explorer:

- a. In the Navigator view, right-click the coordination queue manager and select **Properties**.
- b. From the **Properties** pane, select **Publish/Subscribe**.
- c. Select **Compatibility** from the **Publish/Subscribe mode** list.

MQSC:

```
ALTER QMGR PSMODE(COMPAT)
```

2. Send another message.

Perform the Managed File Transfer action that has publication problems. For example, for agent registration, a message is sent whenever the agent is started (you do not need to repeatedly delete and create the agent to generate registration messages). Because the publish/subscribe engine is disabled, no publication takes place.

3. Browse the SYSTEM.FTE queue on the coordination queue manager.

You should use the IBM MQ Explorer to browse your coordination queue manager's SYSTEM.FTE queue.

IBM MQ Explorer:

- a. In the Navigator view, expand the coordination queue manager and click **Queues**. In the Content view, right-click the SYSTEM.FTE queue and select **Browse Messages**. The **Message browser** window opens and shows the messages that would have been published.
- b. The **User identifier** column shows the user ID contained in the message descriptor. A common reason for publication failure is that this user ID does not have publish authorization on the SYSTEM.FTE topic.
- c. You can find out more information about each message (including the XML that will be published) by right-clicking the message and selecting **Properties**.

There is no MQSC command to inspect the contents of messages. If you do not have the IBM MQ Explorer, you must use a different program that can browse queues and display all aspects of the messages found. You can use the **amqsbcg** sample program, if installed, as described in the following topic: [Browsing queues](#). The `UserIdentifier` line shows the user ID. Alternatively, you can use **dmpmqmsg**; the user ID for a message is found in lines like:

```
A RTM MQ24
A USR JOHND0E
A ACC 1A0FD4D8F2F4C3C8C9D5F1F9C6F7C1C3F3F00019F7AC3000000000000000000
```

The second line in the example is the message descriptor user ID for that message.

4. Re-enable the coordination queue manager publish/subscribe engine.

You can either complete this step using the IBM MQ Explorer or using MQSC commands. After you have re-enabled the publish/subscribe engine in the coordination queue manager, any messages on the SYSTEM.FTE queue are processed immediately.

IBM MQ Explorer:

- a. In the Navigator view, right-click the coordination queue manager and select **Properties**.
- b. From the **Properties** pane, select **Publish/Subscribe**.
- c. Select **Enabled** from the **Publish/Subscribe mode** list.

MQSC:

```
ALTER QMGR PSMODE(ENABLED)
```

IBM i Possible errors when transferring IBM i save files

If you use Managed File Transfer to transfer the same IBM i save file several times, the transfer might fail.

Managed File Transfer might produce one or both of the following errors:

- BFGII0003E: Unable to open file "/qsys.lib/library.lib/SAVF.FILE" for reading
- BFGII0082E: A file open for read failed due to a Java IOException with message text "Sharing violation occurred"

These errors can occur if you issue several concurrent requests for an MFT agent to transfer the same IBM i save file. If you want to concurrently transfer the same save file several times, you must use several source agents. Use a different source agent for each concurrent transfer.

To transfer the same save file several times with a single source agent, you must wait until the previous transfer request is complete before submitting each new transfer request.

V 9.0.1 Possible errors when configuring the Redistributable MFT Agent

Error messages when you are configuring the Redistributable Managed File Transfer Agent

Native library for Windows could not be loaded

Windows You must install the following Microsoft libraries on your system to use the Redistributable Managed File Transfer Agent:

- Microsoft Visual C++ Redistributable 2008
- Microsoft Visual C++ Redistributable 2012

These libraries are available from Microsoft. See [The latest supported Visual C++ downloads](#).

If these libraries are not installed and you try to run MFT commands, an error is reported:

- BFGUB0070E: Internal error: Native library for platform Windows 7 (architecture amd64) could not be loaded because mqmft (Not found in java.library.path).
- BFGCL0043I: Specify the '-h' command line parameter to see more usage information.

Check that the Microsoft libraries are installed. If the libraries are not installed, install them and run the command again.

Use of bindings mode is not supported

The Redistributable Managed File Transfer Agent can only connect to IBM MQ in client mode. If you try run commands in bindings mode, an error is reported:

- BFGCL0408E: Unable to obtain IBM MQ installation information for queue manager '*queue manager name*'. Reason Cannot run program "../bin/dspmqr": error=2, No such file or directory

When you are issuing commands, you must provide the queue manager host, port, name, and channel name.

Guidance for setting MQ attributes and MFT properties associated with message size

You can change IBM MQ attributes and Managed File Transfer properties to affect the behavior of Managed File Transfer when reading or writing messages of various sizes.

If the size of messages being read from a source queue or written to a destination queue exceeds 1048576 bytes (1 MB), you must increase the value of the Managed File Transfer Agent property **maxInputOutputMessageLength** to a value that is greater than or equal to the maximum message size to be read or written.

If the messages on the source queue are greater than 1048576 bytes, you must set the **maxInputOutputMessageLength** property on the source agent. If the messages on the destination queue are greater than 1048576 bytes you must set the **maxInputOutputMessageLength** property on the destination agent. For more information about the **maxInputOutputMessageLength** property, see [Advanced agent properties](#).

- If the queue that the agent is writing to or reading from is local to the agent queue manager, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of the maximum message size of the source or destination queue is greater than or equal to the value of the **maxInputOutputMessageLength** agent property.

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer (which supports a file size of up to 100 MB):

The value of `maxInputOutputMessageLength`

For a message-to-file transfer:

The value of $3 * (\text{maxInputOutputMessageLength}) + 1048576$

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

- If the queue that the agent is writing to is a remote queue, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of each of the following IBM MQ attributes is greater than or equal to the value of the `maxInputOutputMessageLength` agent property:

- The maximum message size of the remote queue manager transmission queue on the agent queue manager
- The maximum message size of the channel from the agent queue manager to the remote queue manager
- The maximum message size of the destination queue on the remote queue manager
- The maximum message size of the remote queue manager

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer (which supports a file size of up to 100 MB):

The value of `maxInputOutputMessageLength`

For a message-to-file transfer:

The value of $3 * (\text{maxInputOutputMessageLength}) + 1048576$

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

If you exceed the value of one of these properties, the agent stops with the following error in the agent event log:

```
BFGUT0002E: An internal error has occurred. Product failure data was captured in file
"FFDC.FTE.20100928170828514.8172766022149157013.log".
BFGSS0025E: An internal error has occurred. The exception is: cc=2 rc=2010 op=put - MQPUT to
SYSTEM.FTE.STATE.agent_name
BFGAG0061E: The agent ended abnormally
```

The following IBM MQ reason codes might be included in this message in the agent event log:

- `rc=2010` This reason code maps to `MQRC_DATA_LENGTH_ERROR` and indicates that the value of the client channel maximum message size was exceeded. To resolve this problem ensure that the client

channel maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

```
3 * (maxInputOutputMessageLength) + 1048576
```

- rc=2030 This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q and indicates that the value of the maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue was exceeded. To resolve this problem ensure that the maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue is greater than or equal to the result of the following calculation:

```
3 * (maxInputOutputMessageLength) + 1048576
```

- rc=2031 This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q_MGR and indicates that the value of the maximum message size of the agent queue manager was exceeded. To resolve this problem ensure that the maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

```
3 * (maxInputOutputMessageLength) + 1048576
```

If you are transferring many small messages

If the average size of the messages that the agent is reading from or writing to a queue is less than 1310 bytes and the agent is reading or writing more than 10000 messages, you must increase the maximum number of uncommitted messages attribute on the queue manager or reduce the amount of data in a checkpoint interval.

When the agent is reading messages from or writing messages to a queue the corresponding **GETs** or **PUTs** are grouped together into transactions. The number of **GETs** or **PUTs** in a transaction is determined by the number required to process all of the data within a checkpoint interval. The approximate amount of the data in a checkpoint interval is determined from agent properties using the following calculation:

```
Checkpoint interval data size (in bytes) = agentCheckpointInterval * agentFrameSize *  
agentWindowSize * agentChunkSize.
```

The default checkpoint data size is $1 * 5 * 10 * 262144$ bytes = 13107200 bytes (12.5MB). The maximum number of uncommitted messages in a transaction that a queue manager supports is controlled by the **MaxUncommittedMsgs** queue manager attribute. The default value of this attribute is 10000 messages. If the average message size is less than approximately 1310 bytes the default maximum number of uncommitted messages is exceeded if there are more than 10000 messages to be written.

If you exceed the **MaxUncommittedMsgs** limit, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2024' from the message queue interface (MQI).  
The agent cannot continue processing and will now end.  
BFGAG0139I: The agent has suspended its current transfers and is now stopping.
```

The reason code 2024 maps to: MQRC_SYNCPOINT_LIMIT_REACHED.

To resolve this problem perform one of the following actions

- Increase the value of the **MaxUncommittedMsgs** queue manager attribute of the queue manager that the agent reading from or writing to a queue connects to. See [MaxUncommittedMsgs \(MQLONG\)](#).
- Reduce the amount of data in a checkpoint interval. To do this, decrease the value of one or more of the following agent properties:
 - agentCheckpointInterval
 - agentFrameSize
 - agentWindowSize

– agentChunkSize

For information about these agent properties, see [Advanced agent properties](#).

If you are writing messages to a queue persistently

If you are transferring to a queue and writing the messages to the queue persistently, you might have to increase the size of the queue manager log file space to be able to log all of the data in a checkpoint interval.

If you exceed the queue manager log file space, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2102' from the message queue interface (MQI).  
The agent cannot continue processing and will now end.  
BFGAG0062E: The agent has received MQI reason code '2102'. The agent cannot continue processing and  
will now end.  
BFGAG0061E: The agent ended abnormally
```

The reason code '2102' maps to: MQRC_RESOURCE_PROBLEM.

To resolve this problem increase the size of the destination agent queue manager log file space.

Guidance for using UAC and virtual store with MFT

User Account Control (UAC) is present in Windows Server 2008 R2 and other similar operating systems. This is a security infrastructure and one of its features is to divert user data stored in the central Program Files directory to a user location, which is known as virtual store.

If only the Managed File Transfer tools are used to manage the data structures, Managed File Transfer is not affected by UAC and virtual store. However, if the directory structure is changed or rebuilt using standard operating system tools by a non-IBM MQ administrator, it is possible the new structure will be diverted into a virtual store. This can cause one or more of the following situations:

- Users, including the IBM MQ administrator, can no longer see files in their expected location.
- An agent might fail to start, reporting message BFGCL0315 but give no supporting reason code.
- The log files cannot be found at the location reported by the agent.
- An agent when started with the **-F** parameter might fail to start, reporting message:

```
The current directory is invalid
```

To correct all of these situations:

- As an IBM MQ administrator, use the **fteDeleteAgent** and **fteCreateAgent** commands to rebuild the agent structure.
- As an operating system administrator, remove the IBM MQ entries in the virtual store of the affected users. For example, on Windows the location of the virtual store is as follows: `%USERPROFILE%\AppData\Local\VirtualStore\`

Related reference

[“fteDeleteAgent” on page 442](#)

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

[“fteCreateAgent \(create an MFT agent\)” on page 371](#)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

Windows Guidance for running an MFT agent or logger as a Windows service

You can run a Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the information in this topic to diagnose the issue.

For information about configuring your agent, stand-alone logger, or stand-alone file logger, to run as a Windows service, see [“Starting an MFT agent as a Windows service” on page 147](#) and [“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#).

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

Location of log files

When you use the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**, or **fteModifyLogger** command to run an agent or logger as a Windows service, you can choose the level of logging by using the **-sl** parameter. The possible values for this parameter are *error*, *info*, *warn*, and *debug*. The default value is *info*.

The log file for the Windows service has the file name *servicedate.log*, where *date* is the date when the service was started. The file for an agent is written to the directory *MQ_DATA_PATH\mqft\logs\coordination_qmgr_name\agents\agent_name*. This directory is the same directory that Managed File Transfer Agent trace files are written to. The file for the logger is written to the directory *MQ_DATA_PATH\mqft\logs\coordination_qmgr_name\loggers\logger_name*.

If you have problems starting an agent, or a stand-alone logger as a Windows service, try setting the logging level to debug using the **-sl** parameter. Additional information is written to the *servicedate.log* file.

Note: When the logging level is set to debug, the user account and password that you are using to run the Windows service are shown in the log file in plain text.

Number of log files

When you use the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**, or **fteModifyLogger** command to run an agent or a stand-alone logger as a Windows service, you can choose the number of log files by using the **-sj** parameter. Specify the following text as part of your command to change the number of log files: **-sj -Dcom.ibm.wmqfte.daemon.windows.windowsServiceLogFiles=number**, where *number* is the number of log files that you want expressed as a positive integer. If you do not specify the number of log files, the default is five.

"Log on as a service" authority

The Windows account that you use to run the service must have the **Log on as a service** right. If you try to start the service, either with the **fteStartAgent**, **fteStartLogger** command, or with the Windows **Sc.exe** command, and you are using a user account that does not have this right, a **Services** window opens. If the service you wanted to start was to run an agent, this window contains the following message:

```
Unable to start Windows service mqmftAgentAGENT@QMGR.  
System error 1069: The service did not start due to a logon failure.
```

In this message, *AGENT* is your agent name and *QMGR* is your agent queue manager name. If you are trying to run a stand-alone logger as a service, a similar message is produced, which refers to the logger rather than an agent.

To prevent this error, give the Windows account that you use to run the service the **Log on as a service** right. For example, on Windows 7 complete the following steps:

1. From the **Start** menu, click **Administrative Tools > Local Security Policy**.
2. In the **Security Settings** pane, expand **Local Policies**, and then click **User Rights Assignments**.
3. In the **Policy and Security Setting** pane, double-click **Log on as a service**.
4. Click **Add User or Group**, and then add the user that you want to run the service to the list of users that have the **Log on as a service** right. You provided this user name when you ran the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**, or **fteModifyLogger** command.

Note: The error System error 1069: The service did not start due to a logon failure. can also be caused by an incorrect password.

Hiding your Windows account password

When you configure your agent or stand-alone logger to run as a Windows service, you specify a user name and password to use. In the following example, the agent AGENT1 is created, which has an agent queue manager QMGR1 and is configured to run as a Windows service:

```
fteCreateAgent -agentName AGENT1 -agentQMGR QMGR1 -s -su fteuser -sp ftepassword
```

In this example, the Windows service runs with a user name of `fteuser`, which has an associated password `ftepassword`. When you run the **fteCreateAgent** command, or one of the other commands that accepts the **-s** parameter, you specify the password for the Windows account in plain text. If you prefer not to display your password, carry out the following steps:

1. Run the command (**fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger** or **fteModifyLogger**) without specifying the **-sp** parameter. For example:

```
fteCreateAgent -agentName AGENT1 -agentQMGR QMGR1 -s -su fteuser
```

Note: The command produces a message that warns you that you must set the password by using the Windows Services tool before the service starts successfully.

2. Open the Windows **Services** window.
3. In the list of services, right-click the agent or stand-alone logger service and select **Properties**. The agent service display name is Managed File Transfer Agent *AGENT* @ *QMGR*, where *AGENT* is the agent name and *QMGR* is your agent queue manager name. The logger service display name is Managed File Transfer Logger for property set *coordination_qmgr_name*, where *coordination_qmgr_name* is the coordination queue manager that you specified for the stand-alone logger to use as its property set. For more information about the property set, see [“fteStartLogger” on page 507](#) and [“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#).
4. In the **Properties** window, select the **Log On** tab.
5. Enter the password for the user account that runs the service in the **Password** and **Confirm password** fields. The password characters are hidden as you enter them.
6. Click **OK**.

Known issues

Problem using the JAVA_HOME system environment variable (applies to Managed File Transfer in IBM WebSphere MQ 7.5.0 Fix Pack 1 or earlier only).

The JAVA_HOME system environment variable must not be set, otherwise the agent or logger Windows Service is unlikely to start. The agent or logger Windows Service must be run with the IBM MQ Java runtime.

Guidance for updating agent or logger JVM options

If you use the **-sj** parameter of the **fteModifyAgent** or **fteModifyLogger** command to modify an existing Windows Service definition for an agent or logger by updating, adding, or removing Java system properties, the existing Windows Service is first deleted before a new one is created in its place, and the agent or logger properties file is updated with the properties for the new Windows Service. The new Windows Service definition must be consistent with the updated Windows Service properties that are defined in the agent or logger properties file.

From IBM MQ 9.0.0 Fix Pack 4, additional checks are added under APAR IT22423 such that any updates that are made to the JVM options for an agent or logger with the **-sj** parameter of the **fteModifyAgent** or **fteModifyLogger** command are verified to make sure that the options have been correctly specified. If the properties are found to be invalid, or otherwise could not be validated, the **fteModifyAgent** or **fteModifyLogger** command fails and an appropriate error message is displayed.

If the JVM properties are valid and the deletion of the existing Windows Service is successful, but a failure then arises when the **fteModifyAgent** or **fteModifyLogger** command is creating the new Windows Service, the command attempts to remove the properties that define the replacement Windows Service from the agent or logger properties file. In this case, error messages are returned to explain that the agent or logger could not be modified, the old Windows Service was deleted but a new Windows Service could not be created and the agent or logger will therefore not run as a Windows Service. You must then manually verify that the state of the Windows Service definition is consistent with the Windows Service properties that are defined in the agent or logger properties file, and take the appropriate action to correct any inconsistencies.

Related reference

[fteModifyAgent: run an MFT agent as a Windows service](#)

[fteModifyLogger: run an MFT logger as a Windows service](#)

Guidance for configuring an MFT resource monitor to avoid overloading an agent

You can configure the property and parameter values of a Managed File Transfer resource monitor to reduce the load on an agent. Reducing the load on the agent improves the performance of that agent. There are several settings you can use, and you may need to use trial and error to find the best settings for your system configuration.

Overview of resource monitoring

When a resource monitor polls a directory or a queue, the agent completes the following stages:

- Finds all the files that match a trigger pattern (for example, all the *.txt files in the directory). Or finds all complete groups of messages on the queue.
- Determines which files are new or changed, or determines which groups are new on the queue.
- Initiates transfers for the files or groups that match the criteria in the two previous stages.
- Adds to the list of files and groups already transferred so they are not transferred again until they change.

For a directory monitor, the more files in the source directory and the broader the triggering pattern, the bigger the list of files the agent has to parse and compare against the list of files already transferred.

For a queue monitor, the more groups on the queue the bigger the list of groups the agent has to compare against the list of groups already transferred.

Consider the following key settings:

- Use agent property **monitorMaxResourcesInPoll** to set the maximum number of files or groups the agent includes on each poll. Using this parameter limits the number of transfers in a polling interval. It also means that the agent has less parsing to do before initiating a transfer for that number of files

or groups. The next time the directory monitor or queue monitor polls, the agent includes the next set of files or groups. Agent property **monitorMaxResourcesInPoll** is available in IBM WebSphere MQ File Transfer Edition 7.0.4 and later, for earlier versions of IBM WebSphere MQ File Transfer Edition it is available as an interim fix for APAR IC78011.

- When creating a directory monitor, ensure that the transfer definition you configure has a source disposition of `delete`. Setting this disposition means that when the file transfer completes it is removed from the monitored directory and the agent no longer keeps it on its internal list.
- When creating a directory monitor, use the **-r1** parameter in the **fteCreateMonitor** command to limit the number of levels of the directory the agent has to recurse through. Using this parameter means that lower-level directories are not scanned unnecessarily.

Further considerations when creating a resource monitor

The process of resource monitor polling consumes agent resources. Increasing the polling interval of a monitor reduces the load placed on the agent. However, the setting of the polling interval must be balanced against generating too many transfers per polling interval. Consider the following when you set the polling interval for a resource monitor:

- How quickly you need a transfer to be initiated after a file is placed in a directory, or a group on a queue.
- The rate which files are placed into a directory, or groups onto a queue.
- The maximum transfer rate of the agent. The agent must be able to handle all the transfers that a monitor generates.

The polling interval is specified when the resource monitor is created with the **fteCreateMonitor** command by specifying the **-pi** (polling interval) and **-pu** (polling interval units) parameters. You may need to experiment to determine the best settings for your configuration.

An option to improve the stability of highly loaded agents that run resource monitors, is to reduce the agent property value of `maxSourceTransfers`. With this option the agent splits its processing time between the resource monitor and transferring files. The higher the value of agent property `maxSourceTransfers`, the more processing time is consumed by transferring files and less is available for the resource monitor. If you reduce the value of agent property `maxSourceTransfers`, the agent does fewer transfers in parallel, but it should have enough processing time to poll its resource monitors. If you lower the value of this agent property you should consider increasing the value of agent property `maxQueuedTransfers` because the number of queued transfers may increase.

If after optimizing your monitor you find that some transfers enter recovery, consider increasing an agent timeout value. The heavy load placed on the agent, may mean that the transfers timeout when negotiating the start of the transfer with the destination agent. This timeout causes the transfer to go into recovery and delays the completion of the transfer. The agent property `maxTransferNegotiationTime` specifies the time the source agent waits for a response from the destination agent. If this time is exceeded the transfer goes into recovery. The default value of this property is 30000 milliseconds (30 seconds). Increasing the value of the property, for example to 300000 Milliseconds (5 minutes), may allow the transfers to continue without timing out and avoid going into recovery.

Related concepts

[“Using transfer definition files” on page 152](#)

You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Related tasks

[“Monitoring MFT resources” on page 162](#)

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

Related reference

[“fteCreateMonitor: create an MFT resource monitor” on page 394](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

What to do if variable substitution causes multiple files to go to a single file name

For Managed File Transfer, if you are monitoring a directory and transferring multiple files from a source to a destination location and you are using `${FileName}` variable substitution, you must test the variable substitution results. The results need to be tested because the use of variable substitution might cause unexpected combinations of file transfer commands to be invoked.

To determine whether the problem is occurring, look for cases of multiple files appearing to transfer but only one file arriving at the destination. You might see errors in the file transfer log showing multiple files attempting to transfer to the same destination file name and failing transfers to the same file name.

Why this problem occurs

When multiple files are being processed by an MFT directory monitor, the Task xml runs for every file that the monitor finds in the directory being monitored. If the `${FileName}` is only specified in the destination of the xml task file and not the source, the transfer is invoked for each file multiple times, once for each file name combination.

For example:

```
<source disposition="delete" recursive="false">
  <file>e:\temp</file>
</source>
<destination exist="overwrite" type="file">
  <file>s:\outdir\${FileName}</file>
</destination>
```

Avoiding this problem

If you are using `${FileName}` variable substitution in the source or destination and are expecting a variation of the same file name to arrive at the destination, be sure to specify `${FileName}` in BOTH the source and destination of your task XML definition.

The following example takes a file from `e:\temp\<filename>` and transfers it to `s:\outdir\<filename>.out`:

```
<source disposition="delete" recursive="false">
  <file>e:\temp\${FileName}</file>
</source>
<destination exist="overwrite" type="file">
  <file>s:\outdir\${FileName}.out</file>
</destination>
```

Related concepts

[“Customizing MFT resource monitor tasks with variable substitution” on page 172](#)

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines that the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

[“Examples: Variable substitution for resource monitor definitions” on page 174](#)

Examples of variable substitution for resource monitor definitions using XML and IBM MQ Explorer.

What to do if you receive an error when updating your MFT database schema on an Oracle database

You might receive the following error message when updating your database schema to the latest level by using the `ftelog_tables_oracle_702_703.sql` file: `ERROR at line 1: ORA-02289: sequence does not exist`. This error occurs because the sequences and triggers used by the tables are not in the same schema as the tables.

About this task

To fix this problem, you must edit the contents of the `ftelog_tables_oracle_702_703.sql` before running it.

Procedure

1. Find out which schema the sequences and triggers used by the Managed File Transfer database logger tables are located in.
 - On Db2, you can use the Control Center to view the tables and schema.
 - On Oracle, you can use the Enterprise Manager to view the tables and schema.
2. Open the `ftelog_tables_oracle_702_703.sql` file in a text editor.
3. In each occurrence of the text `SELECT FTELOG.sequence_name.nextval` replace the text `FTELOG` with the name of the schema where your existing sequences are located.
4. Before each occurrence of the text `CREATE OR REPLACE TRIGGER FTELOG.trigger_name`, insert the text `DROP TRIGGER schema_name.trigger_name`, where `schema_name` is the name of the schema where your existing triggers are located.
5. Use the edited `ftelog_tables_oracle_702_703.sql` file to update the database tables.

MFT logger error handling and rejection

The Managed File Transfer logger identifies two types of error: per-message errors and general errors.

Per-message errors are likely to be caused by a problem with one or a few individual messages. Some examples of situations, which are identified as per-message errors are as follows:

- The result code, which is a required item of data, is missing from a message
- A transfer specifies a job name that is 3000 characters long and too large for the associated database column
- A progress message is received for a transfer, but there is no record of the transfer having been started (perhaps because of a misrouted or delayed transfer start message)
- A message is received, which is not a Managed File Transfer log message

General errors are all those errors that are not per-message errors. These are likely to be because of configuration problems or program errors.

When a per-message error is encountered, the logger rejects the message by placing the message on the reject queue. Nothing is written to the output log, so periodically inspect or continuously monitor the reject queue to detect rejected messages.

If too many messages are rejected consecutively, without any messages being successfully written to the database, this is treated as a general error. For example, consider a site that always uses 10 character codes as job names, but which has inadvertently reconfigured the job name column to be two characters wide. Although data that is too wide is usually a per-message error, in this case the configuration problem is general and is detected as a general error. You can tune the number of consecutive per-message errors needed to cause a general error using the `wmqfte.max.consecutive.reject` property.

If a general error is detected the logger rolls back any messages not yet committed to the queue manager, and then retries periodically. A message identifying the problem is written to the output log and to the console if the logger was started in foreground mode with the **-F** parameter.

The location of the output logs for the logger is dependent on whether it is a stand-alone or JEE database logger. For a stand-alone database logger it is located in the directory `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name`. For a JEE database logger it is located in the standard output log of the application server.

The reject queue

Messages that result in per-message errors are moved to the reject queue. On each rejected message, a message property is set to indicate why the message was rejected. The full name of the property is **usr.WMQFTE_ReasonForRejection**, although `usr.` is omitted in some contexts (including JMS and the IBM MQ Explorer).

If you are using IBM MQ Explorer, you can view the contents of the reject queue by right-clicking the queue and clicking **Browse Messages**. To see why a message was rejected, double-click the message to open its properties dialog, then select the **Named Properties** page. You will see a property called **WMQFTE_ReasonForRejection**. Alternatively, you could write or configure a monitoring tool to obtain this information automatically.

Sometimes, you might want to reprocess messages from the reject queue. In the example described previously in this topic, with a two-character job name column in the database, the messages could be successfully processed after the width of the database column had been increased. As another example, when a transfer-complete message is rejected because its associated transfer-start was missing, the transfer-start message might be received later. Reprocessing the transfer-complete will then be successful.

To reprocess messages, move them from the reject queue to the input queue. In a normal installation, where the logger created its own managed subscription, the input queue is defined by the queue manager and has a name like `SYSTEM.MANAGED.DURABLE.49998CFF20006204`. You can identify the input queue by looking at the **Destination name** in the properties for the subscription `SYSTEM.FTE.DATABASELogger.AUTO`, or using the following MQSC command:

```
DISPLAY SUB(SYSTEM.FTE.DATABASELogger.AUTO) DEST
```

One way of moving messages between queues is to use the [MA01 SupportPac](#), for example:

```
q -IFTE.REJECT -oSYSTEM.MANAGED.DURABLE.49998CFF20006204
```

The reject queue might contain messages rejected for various reasons, only some of which have been resolved. In this case you can still reprocess all the messages; those messages that can now be accepted are consumed, and those messages that cannot are again moved to the reject queue.

Malformed log messages in the transfer log are not logged by the logger. These messages are not viewed as being significant and so these messages are sent to the reject queue. For more information about transfer log messages, see [“File transfer log message formats” on page 723](#).

What to do if the MFT logger is started, but no transfer information is being logged to the database

The database tables used by the Managed File Transfer logger require the database to have a page size of 8 KB or larger. If the page size of the database is not large enough, the tables are not created properly and you see the error `SQLSTATE=42704`.

If you are using the Java Platform, Enterprise Edition database logger, you might see the following message in the WebSphere Application Server system out log; if you are using the stand-alone database logger, you might see the following error in the `output0.log` file:

```
DB2 SQL Error: SQLCODE=-204, SQLSTATE=42704
SQLERRMC=FTELOG.TRANSFER_EVENT, DRIVER=3.40.152
```

The `SQLSTATE` value of 42704 indicates that a table that the logger expected to exist, in this case `FTELOG.TRANSFER_EVENT`, does not exist.

To fix this problem perform the following steps:

1. Check that the table exists and is complete. For information about the tables that the logger uses and their columns, see [“MFT database logger tables”](#) on page 641.
2. If the table does not exist or is incomplete, check the page size of the database.
3. If the database size is less than 8 KB, increase the page size of your database.
 - If your database is on a test system or has no data in it, you can drop the tables and re-create the database with a page size greater than 8 KB.
 - For information about how to increase the page size, see [“Increasing the page size of the log database on Db2 on Windows, UNIX or Linux”](#) on page 19  or [“Migrating the database tables on Db2 on z/OS to IBM MQ 8.0”](#) on page 21.

What to do if MFT does not read keystore properties from the keystore configuration file in AMS

The keystore configuration file location, if not present in the default location, must be specified by the `MQS_KEystore_CONF` variable in order for the Java AMS to run in client mode. If the location is not specified, the Managed File Transfer Agent logs will show the error message: "Failed to read keystore properties from the keystore configuration file."

The default location for the keystore configuration file is `home_directory/.mqc/keystore.conf`. If the location of the keystore configuration file is not the default location, complete the following steps:

1. Start the FTE agent in client mode.
2. Apply AMS security to `SYSTEM.FTE.DATA.<agent name>` queue. If the keystore configuration file is not in this location, all transfers will fail with no acknowledgment.
3. Set the system variable `BFG_JVM_PROPERTIES` to `BFG_JVM_PROPERTIES=-DMQS_KEystore_CONF=path to keystore_config file` for the `fteStartAgent` command.
4. Set the system variable `MQS_KEystore_CONF` to `MQS_KEystore_CONF=path to keystore_config file` for the `fteStartAgent` command. This must be set to ensure all agents run, regardless of the mode they are running in.

Note: If the Java AMS is running in bindings mode, error AMQ9062 will be shown in the queue manager's error log if the keystore configuration file is not in the default location.

What to do if your MFT agent ABENDS with a `java.lang.OutOfMemoryError` due to native memory exhaustion

While processing a number of managed transfer requests, such as file-to-file, message-to-file or file-to-message transfers, the agent abnormally ends (ABENDS) reporting a `java.lang.OutOfMemoryError`,

and at the time your total RAM memory was not fully utilized. This exception has been caused by native memory exhaustion.

Diagnosing the problem

When this issue occurs, the affected agent ABENDs and generates two files that provide details on the root cause:

- An ABEND file. The name of this file conforms to the naming convention `ABEND.FTE.date_timestamp.identifier.log`.

Multi On Multiplatforms, the file is written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/ffdc` directory.

z/OS On z/OS, the file is written to the USS location `$BFG_CONFIG/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/ffdc`

- A Javacore file. The name of this file has the following format: `javacore.datestamp.timestamp.pid.identifier.txt`

Multi On Multiplatforms, the file is written to the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.

z/OS On z/OS, the file is written to the USS location `$BFG_CONFIG/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.

The ABEND and Javacore pair contain information similar to the examples shown below:

Example: Pair one

Abend file

```
Filename:
C:\ProgramData\IBM\MQ\mqft\logs\C0ORDQM\agents\AGENT1\logs\ffdc\ABEND.FTE.20200109113518046.1764802189777
906538.log
Level: p900-005-180821
Time: 09/01/2020 11:35:18:046 GMT
Thread: 96 (TransferSender[414d51204d4424b5250303720202045fbd6532ebfaa02])
Class: com.ibm.wmqfte.thread.FTETHread
Instance: 55b455b4
Method: uncaughtException
Probe: ABEND_001
Cause: java.lang.OutOfMemoryError: native memory exhausted

java.lang.OutOfMemoryError: native memory exhausted
at com.ibm.mq.jmqi.local.internal.base.Native.MQPUT(Native Method)
at com.ibm.mq.jmqi.local.LocalMQ.MQPUT(LocalMQ.java)
at com.ibm.wmqfte.wmqiface.WMQQueueImpl.put(WMQQueueImpl.java)
at com.ibm.wmqfte.wmqiface.WMQQueueImpl.put(WMQQueueImpl.java)
at com.ibm.wmqfte.transfer.impl.TransferSenderRunnable.doTransfer(TransferSenderRunnable.java)
at com.ibm.wmqfte.transfer.impl.TransferSenderRunnable.run(TransferSenderRunnable.java)
at java.lang.Thread.run(Thread.java)
at com.ibm.wmqfte.thread.FTETHread.run(FTETHread.java)
```

Javacore file

```
NULL -----
0SECTION TITLE subcomponent dump routine
NULL =====
1TISIGINFO Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError" "native memory
exhausted" received
1TIDATETIME Date: 2020/01/09 at 11:35:18
1TIFILENAME Javacore filename:
C:\ProgramData\IBM\MQ\mqft\logs\C0ORDQM\agents\AGENT1\javacore.20200109.113518.14148.0002.txt
```

Example: Pair two

ABEND file

```
Filename:
C:\ProgramData\IBM\MQ\mqft\logs\C00RDQM\agents\AGENT1\logs\ffdc\ABEND.FTE.20200109143700286.3177895731698
464509.log
Level: p900-005-180821
Time: 09/01/2020 14:37:00:286 GMT
Thread: 918 (AgentStatusPublisher)
Class: com.ibm.wmqfte.thread.FTEThread
Instance: bc10bc1
Method: uncaughtException
Probe: ABEND_001
Cause: java.lang.OutOfMemoryError: Failed to create a thread: retVal -1073741830, errno 12

java.lang.OutOfMemoryError: Failed to create a thread: retVal -1073741830, errno 12
at java.lang.Thread.startImpl(Native Method)
at java.lang.Thread.start(Thread.java)
```

Javacore file

```
NULL -----
0SECTION TITLE subcomponent dump routine
NULL =====
1TISIGINFO Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError" "Failed to create a
thread: retVal -1073741830, errno 12" received
1TIDATETIME Date: 2020/01/09 at 14:37:00
1TIFILENAME Javacore filename: C
C:\ProgramData\IBM\MQ\mqft\logs\C00RDQM\agents\AGENT1\javacore.20200109.143700.2652.0003.txt
```

Why this problem occurs

This issue occurs due to exhaustion of the native heap memory on the system where the agent is running.

See [“How MFT agents use Java heap and native heap memory”](#) on [page 707](#) for more information on the distinctions between Java heap memory and native heap memory.

Avoiding the problem

There are a number of actions that you can take to help reduce the likelihood of an MFT agent stopping due to a `java.lang.OutOfMemoryError`, caused by exhaustion of native memory:

1. Reduce the size of the Java heap for the JVM running the MFT agent.

The greater the size of the allocated Java heap, the less memory is available to the native heap. Reducing the size of the Java heap used by an agent can free up more memory for the native heap.

By default, the Java heap of an agent is set to 512 MB. If you have changed this to make it a larger value, consider reducing it, and testing with your production-like workload.

- To lower or change the Java heap when running the agent as a normal process:

Set the `BFG_JVM_PROPERTIES` environment variable to pass options directory to the JVM. For example, on Windows, to set the maximum heap size to 1024 MB run the following command before using the **fteStartAgent** command:

```
set BFG_JVM_PROPERTIES="-Xmx1024M"
```

For more information about how to set Java system properties using the `BFG_JVM_PROPERTIES` environment variable, see [Java system properties for MFT](#).

- To lower or change the Java heap when running the agent as a Windows service:

To pass options to the JVM running the agent as a Windows service, modify the agent using the **-sj** parameter specified on the [fteModifyAgent](#) command.

The following example uses the **fteModifyAgent** command with the **-sj** parameter, to set the maximum size of the Java heap for a JVM running a Windows service configured agent:

```
fteModifyAgent.cmd -agentName AGENT1 -s -su user1 -sp password -sj -Xmx1024M
```

You can check this has been successfully set, by reviewing the `output0.log` file of the agent, after the agent has been restarted. In the *Start Display Current Environment* section, a value of 1024 MB will be reported, as follows:

```
The maximum amount of memory that the Java virtual machine will attempt to use is: '1024'MB
```

2. Restrict native memory use

Often, `java.lang.OutOfMemoryErrors` caused by native heap exhaustion are seen if an agent connects to its agent queue manager using the BINDINGS transport. When the agent has been configured to use the BINDINGS transport, the agent calls native methods whenever it needs to communicate with the queue manager.

This means that native memory usage grows as the workload of the agent increases, due to more connections to the queue manager and increased message communication. In this situation, reducing the workload can help. To do this, set the following agent properties to a lower value than the default 25:

- **maxSourceTransfers**
- **maxDestinationTransfers**

This reduces the number of concurrent transfers that can occur, and therefore decreases the maximum concurrent workload for the agent.

3. Configure the agent to use the CLIENT transport when connecting to its agent queue manager. You can do this by setting the following agent properties:

- **agentQMgrHost**
- **agentQMgrPort**
- **agentQMgrChannel**

You can find information about these properties in [The MFT agent.properties file](#) topic.

This ensures that all communication between the agent and the queue manager takes place over TCP/IP, rather than native code, which reduces the amount of native memory used by the agent.

Important: Taking this action also decreases performance. By using a TCP/IP connection to the local host, rather than native code, the configuration is not as efficient when the agent requires interactions with the queue manager.

BFGSS0023E errors and how to avoid them

If you uninstall a Fix Pack from an installation in order to move back to a previous version of the product, and an agent associated with the installation was involved with managed transfers at the time the uninstallation took place, then that agent cannot start and will report an BFGSS0023E error. You can avoid this error by completing a number of steps that should prevent BFGSS0023E messages from appearing when the agents are restarted.

For every in-flight managed transfer that an agent is currently involved in, there is a message on the agent's `SYSTEM.FTE.STATE.agent_name` queue. This message stores checkpoint information on the managed transfer, and is used if the managed transfer goes into recovery. Once a managed transfer has finished, then the corresponding message on the `SYSTEM.FTE.STATE.agent_name` queue is removed.

Each state message contains some internal header information indicating which version of the Managed File Transfer component was being used by an agent when the managed transfer was running. The version information shows the specific Fix Pack level, so, for example, if an IBM MQ 8.0.0 Fix Pack 5 agent was running a managed transfer, then the state message for that managed transfer would contain a reference to IBM MQ 8.0.0 Fix Pack 5.

If a Fix Pack is uninstalled from an installation, and an agent associated with that installation has in-flight transfers associated with it, then the agent fails to start and reports the following error:

```
BFGSS0023E: The agent is configured to use IBM MQ queues that contain data created using a later version of the product. The agent cannot run in this configuration and will end.
```

For example, if an IBM MQ 8.0.0 Fix Pack 5 agent has some in-flight transfers running when it is stopped and then downgraded to the IBM MQ 8.0.0 Fix Pack 4 level, the next time the agent is started, it checks the messages on its `SYSTEM.FTE.STATE.agent_name` queue and finds that they were written when it was using IBM MQ 8.0.0 Fix Pack 5. As it is now using IBM MQ 8.0.0 Fix Pack 4, the agent reports the BFGSS0023E error described in the previous paragraph and shuts itself down.

As a general rule, if you want to remove a Fix Pack to either the Managed File Transfer component completing the following steps should prevent the BFGSS0023E messages from appearing when the agents are restarted:

1. Ensure that all of their agents have completed their managed transfers.
2. Stop the agents.
3. Remove the Fix Pack.
4. Restart the agents.

Related tasks

[“Starting an MFT agent” on page 146](#)

Before you can use a Managed File Transfer agent for a file transfer, you must first start the agent.

[Reverting a queue manager to a previous version on UNIX](#)

[Reverting a queue manager to a previous version on Windows](#)

Related reference

[“MFT Agent queue settings” on page 595](#)

The MQSC command scripts generated by the **fteCreateAgent** command create the agent queues with parameters set to the following values. If you do not use the MQSC scripts provided to create the queues, but create the queues manually, ensure you set the following parameters to the values given.

[BFGSS0001 - BFGSS9999](#)

What to do if managed transfers fail with BFGIO0341E errors

If a managed transfer is transferring a file into a location that is being monitored by an external process, then it is possible for that managed transfer to fail with the error: BFGIO0341E: The rename of temporary file `destination_filename.part` to `destination_filename` failed because the temporary file does not exist. This is due to the way that the destination agent for managed transfers uses temporary files when writing a destination file.

How a destination agent uses temporary files

By default, when a managed file transfer takes place, the destination agent performs the following steps:

- Create a temporary file, called `destination_filename.part`.
- Lock the temporary file.
- Write file data into the temporary file, when it is received from the source agent.
- Unlock the temporary file after all of the file data has been received and written out.
- Rename the temporary file, from `destination_filename.part` to `destination_filename`.

If a managed transfer goes into recovery, then it is possible for the destination agent to create temporary files called `destination_filename.partnumber`. The destination agent then writes the file data to this file, instead of the one called `destination_filename.part`.

If the temporary filename `destination_filename.partnumber` already exists, the destination agent tries to create a new temporary file with the name `destination_filename.part(number + 1)`.

If that file already exists, the destination agent attempts to create a temporary file with the name

destination_filename.part(number + 2), and so on until it is successfully able to create the file. In the situation that the agent tries, and fails, to create the temporary file *destination_filename.part1000*, it writes directly to the destination file and does not use a temporary file.

When a managed transfer completes, the destination agent deletes all of the temporary files that are called *destination_filename.partnumber*, as the assumption is that these were created by the agent during the managed transfer.

Note: If the agent property **doNotUseTempOutputFile** is set to the value true, the destination agent does not use temporary files. Instead, it writes directly to the destination file. For more information about the **doNotUseTempOutputFile** property, see [The MFT agent.properties file](#).

Why this problem occurs

A BFGIO0341E error is generated if the destination agent attempts to rename the temporary file, only to find that file is no longer there. A typical scenario that can cause this problem is as follows:

- A *staging directory* has been set up on the target file system.
- An external process is configured to monitor the *staging directory*, and move any files that it finds to a new location.
- The destination agent creates and locks the temporary file *destination_filename.part* in the *staging directory*.
- The destination agent writes file data into the temporary file.
- After all of the file data has been written to the temporary file, the destination agent unlocks the file.
- The external process finds the temporary file, and moves it to the new location.
- The destination agent attempts to rename the temporary file, and finds that it is no longer there. As a result, the transfer item is marked as **Failed** with a BFGIO0341E error.

Avoiding this problem

There are two ways to prevent the BFGIO0341E error from occurring:

- Temporary files written by a destination agent always end with the *.part* or *.partnumber* suffix. If you can configure the external process to ignore those files rather than moving them, the files will still exist in the target directory when the destination agent performs the rename operation.
- Alternatively, configure the destination agent so that it does not use temporary files, and writes directly to the destination file. The destination file is unlocked only when all of the file data has been written to it, at which point it can be picked up by the external process.

To configure the destination agent to write directly to the destination file, set the agent property **doNotUseTempOutputFile=true**. For more information about this property, see [The MFT agent.properties file](#).

Return codes for MFT

Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

The following table lists the product return codes with their meanings:

Return code	Short name	Description
0	Success	The command was successful
1	Command unsuccessful	The command ended unsuccessfully.

Table 24. Return codes (continued)

Return code	Short name	Description
2	Command timed out	The agent did not reply with the status of the command within a specified timeout. By default, this timeout is unlimited for managed call and transfer commands. For example, when you specify the -w parameter with the fteCreateTransfer command. By default, this timeout is 5 seconds for other commands.
3	Acknowledgement timed out	The agent did not acknowledge receipt of the command within a specified timeout. By default, this timeout is 5 seconds.
4	Wrong agent	The command was sent to the wrong agent. The agent specified in the command XML is not the agent that is reading the command queue, on which the message was placed.
20	Transfer partially successful	The transfer completed with partial success and some files were transferred.
21	Transfer stopped	The transfer was stopped by one of the user exits.
22	Cancel transfer timed out	The agent received a request to cancel a transfer but the cancellation could not be completed within 30 seconds. The transfer was not canceled.
26	Cancel ID not found	The agent received a request to cancel a transfer but the transfer cannot be found. This might be because the transfer completed before the cancel request was processed by the agent. It might also be caused because you supplied an incorrect transfer ID to the fteCancelTransfer command. The cancel request was ignored.
27	Cancel in progress	The agent received a request to cancel a transfer, but the transfer is already in the process of being canceled. The new cancel transfer request was ignored.

Table 24. Return codes (continued)

Return code	Short name	Description
40	Failed	The transfer failed and none of the files specified were transferred.
41	Cancelled	The transfer was canceled.
42	Trigger failed	The transfer did not take place because the transfer was conditional and the required condition was not met.
43	Malformed XML	An XML message was malformed.
44	Source agent capacity exceeded	The source agent did not have sufficient capacity to carry out the transfer.
45	Destination agent capacity exceeded	The destination agent did not have sufficient capacity to carry out the transfer.
46	Source agent maximum number of files exceeded	The number of files being transferred exceeded the limit of the source agent.
47	Destination agent maximum number of files exceeded	The number of files transferred exceeded the limit of the destination agent.
48	Invalid log message attributes	A log message is malformed. This error is an internal error. If you receive this return code contact the IBM support center for further assistance.
49	Destination unreachable	The source agent is unable send a message to the destination agent due to an IBM MQ problem. For example if the source agent queue manager has not been configured correctly to communicate with the destination agent queue manager.
50	Trial version violation	An attempt was made by a trial version agent to communicate with an agent that is not a trial version agent.
51	Source transfer not permitted	The maxSourceTransfers agent property has been set to 0. It is not permitted for this agent to be the source of any transfers.

Table 24. Return codes (continued)

Return code	Short name	Description
52	Destination transfer not permitted	The <code>maxDestinationTransfers</code> agent property has been set to 0. It is not permitted for this agent to be the destination for any transfers.
53	Not authorized	The user is not authorized to perform the operation. See the accompanying message for further details.
54	Authority levels do not match	The <code>authorityChecking</code> agent property value of the source agent and destination agent do not match.
55	Trigger not supported	An attempt has been made to create a transfer with a trigger on a protocol bridge agent. This behavior is not supported.
56	Destination file to message not supported	The destination agent does not support writing the file to a destination queue
57	File space not supported	The destination agent does not support file spaces.
58	File space rejected	The file space transfer was rejected by the destination agent.
59	Destination message to file not supported	The destination agent does not support message-to-file transfers.
64	Both queues disallowed	The source and destination of a transfer is a queue.
65	General data queue error	An error occurred when the Managed File Transfer Agent data queue was accessed.
66	Data queue put authorization error	An error occurred when the Managed File Transfer Agent data queue was accessed. Advanced Message Security is not enabled.
67	Data queue put AMS error	An authorization error occurred when the Managed File Transfer Agent data queue was accessed. Advanced Message Security is enabled.
▶ V9.0.1 ▶ V9.0.1 69	Transfer Recovery Timed out	Recovery of a transfer timed out after the specified <code>transferRecoveryTimeout</code> value.

Table 24. Return codes (continued)

Return code	Short name	Description
70	Agent has ended abnormally	Application has had an unrecoverable problem and is forcibly terminating.
75	Queue manager is unavailable	The application cannot continue because the queue manager for the application is unavailable.
78	Problem with the startup configuration	The application cannot continue because there is a problem with the startup configuration data.
85	Problem with the database server	The application cannot continue because there is a problem with the database (typically only returned by a logger)
100	Monitor substitution not valid	The format of a variable substitution within a monitor task XML script was malformed.
101	Monitor resource incorrect	The number of monitor resource definitions was not valid.
102	Monitor trigger incorrect	The number of monitor trigger definitions was not valid.
103	Monitor task incorrect	The number of monitor task definitions was not valid.
104	Monitor missing	The requested monitor is not present.
105	Monitor already present	The requested monitor is already present.
106	Monitor user exit error	A monitor user exit has generated an error during a resource monitor poll.
107	Monitor user exit canceled	A monitor user exit has requested a transaction to be canceled.
108	Monitor task failed	A monitor task has failed to complete due to error in processing the task.
109	Monitor resource failed	A monitor resource definition cannot be applied to the given resource.
110	Monitor task variable substitution failed	A variable has been specified in a monitor task but no matching name has been found in the metadata. Therefore the variable cannot be substituted with a value.

Table 24. Return codes (continued)

Return code	Short name	Description
111	Monitor task source agent not valid	The source agent of the monitor transfer task does not match the agent of the resource monitor.
112	Monitor task source queue manager not valid	The source agent queue manager of the monitor transfer task does not match the agent queue manager of the resource monitor.
113	Monitor not supported	An attempt has been made to create or delete a resource monitor on a protocol bridge agent. This behavior is not supported.
114	Monitor resource denied	The directory that is scanned by the monitor resource is denied access.
115	Monitor resource queue in use	The monitor resource queue is already open, and is not compatible for input with shared access.
116	Monitor resource queue unknown	The monitor resource queue does not exist on the associated queue manager of the monitor.
118	Monitor resource expression invalid	An error occurred evaluating the XPath expression. The XPath expression is evaluated to access the user defined properties in the header of the message. The message is on a queue which is monitored by the resource monitor.
119	Monitor task source agent queue manager missing	The source agent name or source agent queue manager name is missing from the monitor task definition.
120	Monitor queue not enabled	The monitor resource queue is not enabled.
121	Unexpected error when accessing monitor queue	An unexpected error occurred when accessing the monitor resource queue.
122	Monitor command queue not enabled for context id	The monitor agent command queue is not enabled for set context identification.

The following table lists the product intermediate reply codes with their meanings:

Reply code	Short name	Description
-2	ACK	The request has been received but is pending completion.
-3	PROGRESS	The request is for a number of files and some are still pending completion.

Note:

Reply codes are only present if the process that generates the request supplies a reply queue. These are intermediate replies and Managed File Transfer commands return the final reply code only.

Related reference

[“Return codes for files in a transfer” on page 328](#)

Individual files within a transfer have their own result codes which have different meanings to the overall return code from a command.

Return codes for files in a transfer

Individual files within a transfer have their own result codes which have different meanings to the overall return code from a command.

In a transfer log progress message that has an <action> element set to a value of "progress", each file reported has a <status> element with a resultCode. For example:

```
<action time="2009-11-23T21:28:09.593Z">progress</action>
...
  <status resultCode="1">
    <supplement>BFGI00006E: File &quot;C:\destinationfiles\dest1.doc&quot;
      already exists.</supplement>
  </status>
```

The following table describes the possible values for resultCode:

Result code value	Description
0	Success. The file transferred successfully.
1	Failed. The file failed to transfer. See the <supplement> element for more details of the error.
2	Warning. The file transferred but a warning message has been reported. For example, the source file cannot be deleted although the source disposition is set to delete. See the <supplement> element for more details of the warning.

Troubleshooting the Connect:Direct bridge

Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

- [“Tracing the Connect:Direct bridge” on page 329](#)
- [“Log information for the Connect:Direct bridge” on page 329](#)
- [“Solving permissions issues with Connect:Direct nodes” on page 330](#)

- [“What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly” on page 330](#)
-  [“What to do if transfers to PDS or PDS members through the Connect:Direct bridge are failing” on page 331](#)
-  [“Connect:Direct file paths specified with a double forward slash” on page 331](#)
- [“Increasing the number of concurrent transfers for the Connect:Direct bridge” on page 332](#)
- [“Debugging a Connect:Direct process that is called by a file transfer” on page 333](#)

Tracing the Connect:Direct bridge

You can capture trace from the Connect:Direct node that is part of the Connect:Direct bridge to help with problem determination.

About this task

To enable trace, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the Connect:Direct bridge agent properties file to include the line:

```
cdTrace=true
```

3. Start the Connect:Direct bridge agent.

Results

The trace information is written to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Log information for the Connect:Direct bridge

You can use a Connect:Direct bridge agent to transfer files between MFT agents and Connect:Direct nodes. Log information about the Connect:Direct nodes and processes involved in these transfers is displayed in the IBM MQ Explorer plug-in and is stored in your log database.

The Connect:Direct bridge agent must be IBM WebSphere MQ File Transfer Edition 7.0.4 or later. The other agent involved in the transfer can be any version of Managed File Transfer. However, for information about Connect:Direct nodes and processes to be logged, all MFT agents involved in the transfer must be IBM WebSphere MQ File Transfer Edition 7.0.4 or later. For this information to be displayed in the IBM MQ Explorer plugin, the plugin must be IBM WebSphere MQ File Transfer Edition 7.0.4 or later. For this information to be stored in the log database, the database logger and database schema must be IBM WebSphere MQ File Transfer Edition 7.0.4 or later.

Log information about the Connect:Direct nodes and Connect:Direct processes involved in a file transfer is included in the log messages that are published to the `SYSTEM.FTE` topic on the coordination queue manager. For more information, see [“File transfer log message formats” on page 723](#).

The following information is included in the published message:

- Connect:Direct bridge node name
- Primary node (PNODE) name

- Secondary node (SNODE) name
- Process name
- Process ID number

The Connect:Direct bridge node is the same node as either the primary node or the secondary node.

The value of the Connect:Direct bridge node name is the name that the bridge node is known to the MFT Connect:Direct bridge agent by. The primary and secondary node names are the names that are used to refer to the nodes in the network map of the Connect:Direct bridge node.

Related reference

[“Connect:Direct bridge transfer log message examples” on page 741](#)

The destinationAgent or sourceAgent element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

Solving permissions issues with Connect:Direct nodes

Use the information in this topic if your transfers between Managed File Transfer and Connect:Direct fail with an error about insufficient permissions.

For transfers involving the Connect:Direct bridge, the user ID that connects to the Connect:Direct node is determined by which IBM MQ Message Descriptor (MQMD) user ID is associated with the transfer request. You can map specific MQMD user IDs to specific Connect:Direct user IDs. For more information, see [“Mapping credentials for Connect:Direct” on page 138](#).

You might see transfers failing with one of the following errors:

- BFGCD0001E: This task was rejected by the Connect:Direct API with the following error message: Connect:Direct Node detected error.
LCCA000I The user has no functional authority to issue the selp command
- BFGCD0026I: Connect:Direct messages: The submit of the process succeeded. Process number 1092 (name F35079AE, SNODE MYNODE) executing. User fteuser does not have permission to override SNODEID. User fteuser does not have permission to override SNODEID. User fteuser does not have permission to override SNODEID.

If you see either of these errors, determine which Connect:Direct user ID is associated with the MQMD user ID that was used for the transfer request. This Connect:Direct user ID must have authority to perform the Connect:Direct operations required by the Connect:Direct bridge. For the list of functional authorities needed, and guidance on how to grant these authorities, see [“Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file” on page 138](#).

What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly

When you transfer files in text mode between an MFT agent and a Connect:Direct node, code page and end-of-line character conversion is performed. The transfer uses the operating system information in the network map of the Connect:Direct bridge node to determine the end-of-line characters of a remote node. If the information in the network map is incorrect, the end-of-line character conversion might be performed incorrectly.

Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.

- If your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list.
 - If the remote node is on a Windows system, select Windows.

- If the remote node is on a UNIX or Linux system, select UNIX.
-  If the remote node is on a z/OS system, select OS/390.

Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

- Ensure that for each remote node you transfer a file to or from, you specify the operating system type of the remote Connect:Direct node in the `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory. For more information, see “Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes” on page 137 and “Connect:Direct node properties file format” on page 563.

Related reference

“Transferring text files between Connect:Direct and MFT” on page 619

Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between a Managed File Transfer Agent and a Connect:Direct node.

What to do if transfers to PDS or PDS members through the Connect:Direct bridge are failing

If the destination of a transfer is a Connect:Direct node on z/OS and is a PDS or PDS member, the transfer fails if the **-de** parameter has not been specified with a value of overwrite.

About this task

If you submitted the transfer by using the **fteCreateTransfer** or **fteCreateTemplate** command, perform the following steps:

Procedure

1. Change the command that you submitted to include **-de** `overwrite`.
2. Submit the command again.

Using the IBM MQ Explorer plugin

About this task

If you submitted the transfer by using the IBM MQ Explorer plugin, perform the following steps:

Procedure

1. Specify the source and destination information in the **Create New Managed File Transfer** wizard.
2. Select **Overwrite files on the destination file system that have the same name**.
3. Submit the command again.

Connect:Direct file paths specified with a double forward slash

If, as part of a file transfer, you specify a file located on a Connect:Direct node by using a file path that starts with a double forward slash (`//`), the file is treated as a data set.

Sources and destinations on a Connect:Direct node are specified in the format `cd_node_name:file_path`. If the `file_path` starts with a double forward slash (`//`), the source or destination is treated as a data set. This is the case even when the Connect:Direct node is not on z/OS. This can cause transfer failures if the file path is accidentally specified with a double forward slash (`//`) at the start and the file is not a data set.

Ensure that you do not specify a `file_path` that starts with a double forward slash (`//`) if you do not want the file that you specify to be treated as a data set.

Related concepts

[“Troubleshooting the Connect:Direct bridge” on page 328](#)

Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

Related reference

[“Transferring data sets to and from Connect:Direct nodes” on page 609](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

Increasing the number of concurrent transfers for the Connect:Direct bridge

To increase the number of concurrent transfers that the Connect:Direct bridge agent can process, you must change three agent properties. You must also increase the maximum number of connections that the Connect:Direct node accepts.

The maximum number of concurrent transfers that a Connect:Direct bridge agent can process depends on the values of certain agent properties. The **maxSourceTransfers** and **maxDestinationTransfers** agent properties have a default value of five transfers for a Connect:Direct bridge agent. This default value is lower than the default of 25 transfers for other types of agent. A Connect:Direct bridge, where the agent is configured with the default values of **maxSourceTransfers** and **maxDestinationTransfers**, can process a maximum of 10 transfers at any one time: five transfers where the agent is the source, and five transfers where the agent is the destination.

These default values ensure that the Connect:Direct bridge agent does not exceed the maximum number of API connections to the Connect:Direct node. A Connect:Direct bridge agent with the default configuration uses a maximum of 10 API connections to the Connect:Direct node. The maximum number of connections accepted by a Connect:Direct node on UNIX is controlled by the **api.max.connects** Connect:Direct parameter. For a Connect:Direct node on Windows, the equivalent parameter is **max.api.connects**.

If the rate at which your Connect:Direct bridge carries out large numbers of file transfers is not sufficient, you can increase the number of concurrent transfers that the Connect:Direct bridge agent processes. Change the following agent properties for the Connect:Direct bridge agent:

maxSourceTransfers

Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

maxDestinationTransfers

Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

ioThreadPoolSize

The default value of **ioThreadPoolSize** is 10. This property restricts the number of Connect:Direct node API connections for transfers where the Connect:Direct bridge agent is the source agent. These transfers are from Connect:Direct to Managed File Transfer. Use the following guidance to set the value of this property:

- If the value of **maxSourceTransfers** is smaller than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to double the value of **maxSourceTransfers** or 10, whichever is the larger
- If the value of **maxSourceTransfers** is larger than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to the sum of **maxSourceTransfers** and **maxDestinationTransfers**

In addition to these agent properties, you must also change the maximum number of concurrent API connections for the Connect:Direct node that is part of the Connect:Direct bridge. The Connect:Direct parameter that controls this number is **api.max.connects** if your node is on UNIX,

or **max.api.connects** if your node is on Windows. Make the following changes to the appropriate parameter:

api.max.connects (if the node in your Connect:Direct bridge is on UNIX)

Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **api.max.connects** parameter is 16. For more information about how to set this parameter, see the Connect:Direct documentation.

max.api.connects (if the node in your Connect:Direct bridge is on Windows)

Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **max.api.connects** parameter is 10. For more information about how to set this parameter, see the Connect:Direct documentation.

Related tasks

[“Configuring the Connect:Direct bridge” on page 135](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Debugging a Connect:Direct process that is called by a file transfer

You can configure the Connect:Direct bridge agent to write log information about the Connect:Direct process that is called by a file transfer to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

About this task

To configure logging of the Connect:Direct processes, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the `agent.properties` file in the `MQ_DATA_PATH/mqft/config/coordination_queue_manager/agents/bridge_agent_name` directory to include the property `logCDProcess`.
The `logCDProcess` property can have one of the following values:
 - None - No information is logged. This is the default.
 - Failures - Information about failed Connect:Direct processes is logged.
 - All - Information about all Connect:Direct processes is logged.
3. Start the Connect:Direct bridge agent.

Results

Information about Connect:Direct processes is logged to the Connect:Direct bridge agent's `output0.log` file. The information that is logged comprises:

- MFT transfer ID
- Connect:Direct process name
- Connect:Direct process number
- Generated process definition

- File name of the process template, if the Connect:Direct process is user-defined

Related concepts

“Troubleshooting the Connect:Direct bridge” on page 328

Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

Related reference

“The MFT agent.properties file” on page 526

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

MFT reference

MFT security reference

File system permissions for MFT in IBM MQ

When you install and configure the Managed File Transfer component of IBM MQ, the `configuration`, `installations`, and `logs` directories are created with the following permissions.

UNIX and Linux



Table 27. Summary of permissions for directories on UNIX and Linux

Directory	Permissions
<code>/var/mqm/mqft/config</code>	<ul style="list-style-type: none"> • Writable by the mqm group • World readable <p>Users in the mqm group have write access to these directories and files</p>
<code>/var/mqm/mqft/installations</code>	<ul style="list-style-type: none"> • Writable by the mqm group • World readable
<code>/var/mqm/mqft/logs</code>	World readable and writable

Windows



Table 28. Summary of permissions for directories on Windows

Directory	Permissions
<code>MQ_DATA_PATH\mqft\config</code>	<p>The following users have full read and write access:</p> <ul style="list-style-type: none"> • Administrators • System account • mqm group <p>Other users have read access</p>

Table 28. Summary of permissions for directories on Windows (continued)

Directory	Permissions
<code>MQ_DATA_PATH\mqft\installations</code>	The following users have full read and write access: <ul style="list-style-type: none"> • Administrators • System account • mqm group Other users have read access
<code>MQ_DATA_PATH\mqft\logs</code>	The following users have full read and write access: <ul style="list-style-type: none"> • Administrators • System account • mqm group Other users have read and write access

z/OS



Table 29. Summary of permissions for directories on z/OS

Directory	Permissions
<code>DATA_PATH/mqft/config</code>	<ul style="list-style-type: none"> • Writable by the mqm group, or the group name identified in the environment variable BFG_GROUP_NAME • World readable Users in the mqm group, or the value in the environment variable BFG_GROUP_NAME, have write access to these directories and files
<code>DATA_PATH/mqft/installations</code>	<ul style="list-style-type: none"> • Writable by the mqm group, or the group name identified in the environment variable BFG_GROUP_NAME • World readable
<code>DATA_PATH/mqft/logs</code>	World readable and writable

Note: `DATA_PATH` is derived from the environment variable `BFG_DATA`.

Managing authorities for MFT-specific resources

For any file transfer request, the Managed File Transfer Agent processes require some level of access to their local file systems. In addition, both the user identifier associated with the agent process, and the user identifiers associated with users performing file transfer operations must have the authority to use certain IBM MQ objects.

Commands are issued by users, who might be in an operational role where they typically start a file transfer. Alternatively, they might be in an administrative role where they can additionally control when agents are created, started, deleted, or cleaned (that is, when messages from all agent system queues

are removed). Messages containing command requests are placed on an agent's SYSTEM.FTE.COMMAND queue when a user issues a command. The agent process retrieves messages containing command requests from the SYSTEM.FTE.COMMAND queue. The agent process also uses four other system queues, which are as follows:

- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

Because users issuing commands use the queues listed previously in different ways to the agent process, assign different IBM MQ authorities to the user identifiers or user groups associated with each. See [“Restricting group authorities for MFT-specific resources” on page 337](#) for more information.

The agent has additional queues that can be used to grant users the authority to perform certain actions. See [“Restricting user authorities on MFT agent actions” on page 342](#) for information about how to use the authority queues. The agent does not put or get messages on these queues. However, you must ensure that the queues are assigned the correct IBM MQ authorities both for the user identifier used to run the agent process as well as the user identifiers associated with users who are being authorized to perform certain actions. The authority queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

If you are migrating from a version of Managed File Transfer earlier than 7.0.2 to IBM WebSphere MQ 7.5, or later, and are keeping existing agent configurations, you will need to create the authority queues manually. Use the following MQSC command to create the queues:

```
DEFINE QLOCAL(authority_queue_name) DEFPRTY(0) DEFSOPT(SHARED) GET(ENABLED) MAXDEPTH(0) +
  MAXMSGL(0) MSGDLVSQ(PRIORITY) PUT(ENABLED) RETINTVL(999999999) SHARE NOTRIGGER +
  USAGE(NORMAL) REPLACE
```

The agent process also publishes messages to the SYSTEM.FTE topic on the coordination queue manager using the SYSTEM.FTE queue. Depending on whether the agent process is in the role of the source agent or destination agent, the agent process might require authority to read, write, update, and delete files.

You can create and modify authority records for IBM MQ objects using the IBM MQ Explorer. Right-click the object and then click **Object Authorities > Manage Authority Records**. You can also create authority records using the **setmqaut** command, which is described at [setmqaut \(grant or revoke authority\) command](#).

Related reference

[“Restricting group authorities for MFT-specific resources” on page 337](#)

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

[“Restricting user authorities on MFT agent actions” on page 342](#)

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

[“Authorities for the MFT logger” on page 345](#)

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

Restricting group authorities for MFT-specific resources

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

Authority to connect to queue managers

Commands that are run by operational users, administrative users, and the IBM MQ Explorer need to be able to connect to the command queue manager and coordination queue manager. The agent process and commands that are run to create, alter, or delete the agent need to be able to connect to the agent queue manager.

- Grant the FTEUSER group connect authority for the command queue manager and coordination queue manager. For example:

ULW For UNIX, Linux, and Windows systems:

```
setmqaut -m command_queue_manager -t qmgr -g FTEUSER +connect
setmqaut -m coordination_queue_manager -t qmgr -g FTEUSER +connect
```

IBM i For IBM i:

```
GRTMQMAUT OBJ('command_queue_manager') OBJTYPE(*MQM) USER(FTEUSER) AUT(*CONNECT)
GRTMQMAUT OBJ('coordination_queue_manager') OBJTYPE(*MQM) USER(FTEUSER) AUT(*CONNECT)
```

z/OS For z/OS:

```
RDEFINE MQCONN command_queue_manager.BATCH UACC(NONE)
PERMIT command_queue_manager.BATCH CLASS(MQCONN) ID(FTEUSER) ACCESS(READ)
RDEFINE MQCONN coordination_queue_manager.BATCH UACC(NONE)
PERMIT coordination_queue_manager.BATCH CLASS(MQCONN) ID(FTEUSER) ACCESS(READ)
```

- Grant the FTEAGENT group connect and inquire authority to the agent queue manager. For example:

ULW For UNIX, Linux, and Windows systems:

```
setmqaut -m agent_queue_manager -t qmgr -g FTEAGENT +connect +inq +setid
```

IBM i For IBM i:

```
GRTMQMAUT OBJ('agent_queue_manager') OBJTYPE(*MQM) USER(FTEAGENT) AUT(*CONNECT)
```

z/OS For z/OS:

```
RDEFINE MQCONN agent_queue_manager.BATCH UACC(NONE)
PERMIT agent_queue_manager.BATCH CLASS(MQCONN) ID(FTEAGENT) ACCESS(READ)
```

For information about which command directly connects to which queue manager, see [“Which MFT commands and processes connect to which queue manager”](#) on page 349

Authority to put a message on the COMMAND queue that belongs to the agent

The agent command queue must be available to any user who is authorized to request that the agent performs an action. To satisfy this requirement,

- Grant the FTEUSER group only put access to the SYSTEM.FTE.COMMAND.agent_name queue. For example:

ULW**For UNIX, Linux, and Windows systems:**

```
setmqaut -m QM1 -n SYSTEM.FTE.COMMAND.agent_name -t queue -g FTEUSER +put
```

IBM i**For IBM i:**

```
GRTMQMAUT OBJ('SYSTEM.FTE.COMMAND.agent_name') OBJTYPE(*Q) USER(FTEUSER) AUT(*PUT)
MQMNAME('QM1')
```

z/OS**For z/OS:**

```
RDEFINE MQQUEUE QM1.SYSTEM.FTE.COMMAND.agent_name UACC(NONE)
PERMIT QM1.SYSTEM.FTE.COMMAND.agent_name CLASS(MQQUEUE) ID(FTEUSER) ACCESS(UPDATE)
```

- Grant the FTEAGENT group put, get, and setid access to the SYSTEM.FTE.COMMAND.agent_name queue. For example:

ULW**For UNIX, Linux, and Windows systems:**

```
setmqaut -m QM1 -n SYSTEM.FTE.COMMAND.agent_name -t queue -g FTEAGENT +browse +put +get
+setid
```

IBM i**For IBM i:**

```
GRTMQMAUT OBJ('SYSTEM.FTE.COMMAND.agent_name') OBJTYPE(*Q) USER(FTEAGENT) AUT(*PUT)
MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE.COMMAND.agent_name') OBJTYPE(*Q) USER(FTEAGENT) AUT(*GET)
MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE.COMMAND.agent_name') OBJTYPE(*Q) USER(FTEAGENT) AUT(*SETID)
MQMNAME('QM1')
```

z/OS**For z/OS:**

```
RDEFINE MQQUEUE QM1.SYSTEM.FTE.COMMAND.agent_name UACC(NONE)
PERMIT QM1.SYSTEM.FTE.COMMAND.agent_name CLASS(MQQUEUE) ID(FTEAGENT) ACCESS(UPDATE)
RDEFINE MQADMIN QM1.CONTEXT.SYSTEM.FTE.COMMAND.agent_name UACC(NONE)
PERMIT QM1.CONTEXT.SYSTEM.FTE.COMMAND.agent_name CLASS(MQADMIN) ID(FTEAGENT)
ACCESS(UPDATE)
```

Agents need access to put messages to other agents' command queues. If there are agents connected to remote queue managers, you might need to grant additional authorization to allow the channel to put messages to this queue.

Authority to put messages on the DATA, STATE, EVENT, and REPLY queues that belong to the agent

Only Managed File Transfer agents need to be able to use these system queues, therefore grant the group FTEAGENT put, get and inquire access. The names of these system queues are as follows:

- DATA - SYSTEM.FTE.DATA.agent_name
- STATE - SYSTEM.FTE.STATE.agent_name
- EVENT - SYSTEM.FTE.EVENT.agent_name
- REPLY - SYSTEM.FTE.REPLY.agent_name

For example, for the SYSTEM.FTE.DATA.agent_name queue, use a command like the following:

ULW**For UNIX, Linux, and Windows systems:**

```
setmqaut -m QM1 -n SYSTEM.FTE.DATA.agent_name -t queue -g FTEAGENT +put +get +inq
```

For IBM i:

```
GRTMQMAUT OBJ('SYSTEM.FTE.DATA.agent_name') OBJTYPE(*Q) USER(FTEAGENT) AUT(*PUT)
MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE.DATA.agent_name') OBJTYPE(*Q) USER(FTEAGENT) AUT(*GET)
MQMNAME('QM1')
```

For z/OS:

```
RDEFINE MQQUEUE QM1.SYSTEM.FTE.DATA.agent_name UACC(NONE)
PERMIT QM1.SYSTEM.FTE.DATA.agent_name CLASS(MQQUEUE) ID(FTEAGENT) ACCESS(UPDATE)
```

Agents need access to put messages to other agents' data and reply queues. If there are agents connected to remote queue managers, you might need to grant additional authorization to allow the channel to put messages to these queues.

Authority that the agent process runs under

The authority that the agent process runs under affects the files the agent can read and write from the file system, and the queues and topics the agent can access. How the authority is configured is system-dependent. Add the user ID that the agent process runs under to the FTEAGENT group. For more information about adding a user ID to a group, see [Setting up security](#) and navigate to the information for your operating system.

Authority that the commands and IBM MQ Explorer run under

Administrative commands, for example the **fteStartAgent** command, and the Managed File Transfer plug-in for the IBM MQ Explorer need to be able to put messages to the SYSTEM.FTE.COMMAND.*agent_name* queue and retrieve published information from that queue. Add the user IDs that are authorized to run the commands or the IBM MQ Explorer to the FTEUSER group. This originator user ID is recorded in the transfer log. For more information about adding a user ID to a group, see [Setting up security](#) and navigate to the information for your operating system.

Authority to put messages on the SYSTEM.FTE queue and SYSTEM.FTE topic

Only the agent process needs to be able to place messages on the SYSTEM.FTE queue and SYSTEM.FTE topic. Grant put, get and inquire authority to the FTEAGENT group on the SYSTEM.FTE queue, and grant publish and subscribe authority to the FTEAGENT group on the SYSTEM.FTE topic. For example:

For UNIX, Linux, and Windows systems:

```
setmqaut -m QM1 -n SYSTEM.FTE -t queue -g FTEAGENT +put +get +inq
setmqaut -m QM1 -n SYSTEM.FTE -t topic -g FTEAGENT +pub +sub +resume
```

For IBM i:

```
GRTMQMAUT OBJ('SYSTEM.FTE') OBJTYPE(*Q) USER(FTEAGENT) AUT(*PUT) MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE') OBJTYPE(*Q) USER(FTEAGENT) AUT(*GET) MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE') OBJTYPE(*TOPIC) USER(FTEAGENT) AUT(*PUB) MQMNAME('QM1')
GRTMQMAUT OBJ('SYSTEM.FTE') OBJTYPE(*TOPIC) USER(FTEAGENT) AUT(*SUB) MQMNAME('QM1')
```

For z/OS:

```
RDEFINE MQQUEUE QM1.SYSTEM.FTE UACC(NONE)
PERMIT QM1.SYSTEM.FTE CLASS(MQQUEUE) ID(FTEAGENT) ACCESS(UPDATE)
RDEFINE MXTOPIC QM1.PUBLISH.SYSTEM.FTE UACC(NONE)
PERMIT QM1.PUBLISH.SYSTEM.FTE CLASS(MXTOPIC) ID(FTEAGENT) ACCESS(UPDATE)
```

If there are agents connected to remote queue managers, additional authorization might also need to be granted to allow the channel to put messages to the SYSTEM.FTE queue.

For a message to get published to the SYSTEM.FTE topic, the authority records of the SYSTEM.FTE topic must allow publication by the user ID contained in the message descriptor structure (MQMD) of the message. This is described in [Authority to publish log and status messages](#).

To allow a user to publish to the SYSTEM.FTE topic on z/OS, you must grant the channel initiator user ID access to publish to the SYSTEM.FTE topic. If the RESLEVEL security profile causes two user IDs to be checked for the channel initiator connection, you also need to grant access to the user ID contained in the message descriptor structure (MQMD) of the message. For more information, see [The RESLEVEL security profile](#)

Authority to receive publications on the SYSTEM.FTE topic

Transfer log messages, progress messages, and status messages are intended for general use, so grant the FTEUSER group authority to subscribe to the SYSTEM.FTE topic. For example:

ULW

For UNIX, Linux, and Windows systems:

```
setmqaut -m QM1 -n SYSTEM.FTE -t topic -g FTEUSER +sub
```

IBM i

For IBM i:

```
GRTMQMAUT OBJ('SYSTEM.FTE') OBJTYPE(*TOPIC) USER(FTEUSER) AUT(*SUB) MQMNAME('QM1')
```

z/OS

For z/OS:

```
RDEFINE MXTOPIC QM1.SUBSCRIBE.SYSTEM.FTE UACC(NONE)  
PERMIT QM1.SUBSCRIBE.SYSTEM.FTE CLASS(MXTOPIC) ID(FTEUSER) ACCESS(ALTER)
```

Authority to connect to remote queue managers using transmission queues

In a topology of multiple queue managers, the agent requires put authority on the transmission queues used to connect to the remote queue managers.

Authority to create a temporary reply queue for file transfers

File transfer requests wait for the transfer to complete and rely on a temporary reply queue being created and populated. Grant the FTEUSER group DISPLAY, PUT, GET, and BROWSE authorities on the temporary model queue definition. For example:

ULW

For UNIX, Linux, and Windows systems:

```
setmqaut -m QM1 -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g FTEUSER +dsp +put +get +browse
```

IBM i

For IBM i:

```
GRTMQMAUT OBJ('SYSTEM.DEFAULT.MODEL.QUEUE') OBJTYPE(*Q) USER(FTEUSER) AUT(*ADM DSP)  
MQMNAME('QM1')  
GRTMQMAUT OBJ('SYSTEM.DEFAULT.MODEL.QUEUE') OBJTYPE(*Q) USER(FTEUSER) AUT(*PUT)  
MQMNAME('QM1')  
GRTMQMAUT OBJ('SYSTEM.DEFAULT.MODEL.QUEUE') OBJTYPE(*Q) USER(FTEUSER) AUT(*GET)  
MQMNAME('QM1')  
GRTMQMAUT OBJ('SYSTEM.DEFAULT.MODEL.QUEUE') OBJTYPE(*Q) USER(FTEUSER) AUT(*BROWSE)  
MQMNAME('QM1')
```

z/OS

For z/OS:

```
RDEFINE MQQUEUE QM1.SYSTEM.DEFAULT.MODEL.QUEUE UACC(NONE)  
PERMIT QM1.SYSTEM.DEFAULT.MODEL.QUEUE CLASS(MQQUEUE) ID(FTEUSER) ACCESS(UPDATE)
```

By default, this queue is SYSTEM.DEFAULT.MODEL.QUEUE, but you can configure the name by setting values for the properties 'modelQueueName' and 'dynamicQueuePrefix' in the command.properties file.

On z/OS, you must also grant authority to access the temporary queues to FTEUSER. For example:

```
RDEFINE MQQUEUE QM1.WMQFTE.** UACC(NONE)
PERMIT QM1.WMQFTE.** CLASS(MQQUEUE) ID(FTEUSER) ACCESS(UPDATE)
```

By default the name of each temporary queue on z/OS starts with WMQFTE.

The following table summarizes the access control configuration for FTEUSER and FTEAGENT in the security scheme described:

<i>Table 30. Summary of access control configuration for FTEUSER and FTEAGENT</i>			
Object	Object type	FTEUSER	FTEAGENT
Agent queue manager	Queue manager		CONNECT, INQ, and SETID. ALT_USER is also required to enable user authority checking .
Coordination queue manager	Queue manager		
Command queue manager	Queue manager	CONNECT	CONNECT
SYSTEM.FTE	Local queue		GET and PUT
SYSTEM.FTE.COMMAND.agent_name	Local queue	PUT	BROWSE, GET, PUT, and SETID
SYSTEM.FTE.DATA.agent_name	Local queue		GET and PUT
SYSTEM.FTE.EVENT.agent_name	Local queue		BROWSE, GET and PUT
SYSTEM.FTE.REPLY.agent_name	Local queue		GET and PUT
SYSTEM.FTE.STATE.agent_name	Local queue		BROWSE, GET, INQ, and PUT
SYSTEM.FTE	Local topic	SUBSCRIBE	PUBLISH and SUBSCRIBE
SYSTEM.DEFAULT.MODEL.QUEUE (or the model queue defined in Managed File Transfer that is used to create a temporary reply queue.)	Model queue	BROWSE, DISPLAY, GET, and PUT	BROWSE, DISPLAY, GET, and PUT
Transmission queues to communicate with remote queue managers	Local queue		PUT

Authority to manage transfers through IBM MQ Explorer

In addition to granting MFT authorities to users in situations that are already mentioned on this page, further authorities need to be granted to the MFT agent user who administers and performs all MFT operations through IBM MQ Explorer. To issue commands such as create, cancel, schedule file transfer, create, delete resource monitors, and create transfer templates, the IBM MQ Explorer user must have authority as follows:

- Coordination queue manager: connect, inquire, display
- Command queue manager: connect, inquire, display
- SYSTEM.FTE topic: publish, subscribe
- SYSTEM.MQEXPLORER.REPLY.MODEL: display, inquire, get, browse, put

- SYSTEM.ADMIN.COMMAND.QUEUE: inquire, put, display
- SYSTEM.DEFAULT.MODEL.QUEUE: get, put, inquire, display, browse

For information about which command directly connects to which queue manager, see [“Which MFT commands and processes connect to which queue manager”](#) on page 349

Related reference

[“Restricting user authorities on MFT agent actions”](#) on page 342

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

[“Authorities for the MFT logger”](#) on page 345

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

Restricting user authorities on MFT agent actions

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

Enabling user authority management

To turn on user authority checking on agent actions, complete the following steps:

1. In the `agent.properties` file, set the `authorityChecking` value to `true`. For more information, see [“The MFT agent.properties file”](#) on page 526.
2. Ensure that the user who runs the agent has the IBM MQ alternate user (`ALT_USER`) authority to the agent queue manager.

 On the z/OS platform, the user that runs the agent must have `ALT_USER` authority to the user IDs that can request permission to perform an agent action.

Both agents involved in a transfer must have the same level of security enabled, that is, `authorityChecking` must be set to the same value in the property files of both agents. Transfers between agents that have different values for the `authorityChecking` property will fail.

Agent authority queues

The agent has authority queues that are used to manage which users have the authority to perform certain agent actions. The agent does not put or get messages to these queues. The agent authority queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

When user authority management is enabled by setting the agent property **`authorityChecking=true`**, the authorities that a user has on the agent authority queues specify the actions that the user is authorized to take.

Important:  From IBM MQ 9.0.0 Fix Pack 3 and IBM MQ 9.0.4, `inquire` is a required permission on all of the agent authority queues.

The following table summarizes the IBM MQ access authorities that users or groups require in addition to `inquire` permission on an agent authority queue to perform specific actions.

Table 31. The level of IBM MQ access authority that a user or group requires on an agent authority queue to perform specific actions.

User action	Managed File Transfer access authority	Authority queues	IBM MQ access authority (Multiplatforms)	RACF access level (z/OS only)
Shut down the agent, using the -m option on fteStopAgent command.	Administration	SYSTEM.FTE.AUTHADM1. <i>agent_name</i>	BROWSE	READ
Start a transfer of files from this agent	Transfer source	SYSTEM.FTE.AUTHTRN1. <i>source_agent_name</i>	BROWSE	READ
Run a managed call on this agent				
Cancel a transfer of files from this agent started by the same user				
Start a transfer of files to this agent	Transfer destination	SYSTEM.FTE.AUTHTRN1. <i>destination_agent_name</i>	PUT	UPDATE
Cancel a transfer of files to this agent started by the same user				
Create a resource monitor	Monitor	SYSTEM.FTE.AUTHMON1. <i>monitor_agent_name</i>	BROWSE	READ
Delete a resource monitor created by the same user				
Delete a resource monitor created by any user	Monitor operations	SYSTEM.FTE.AUTHOPS1. <i>agent_name</i>	SET	ALTER
Create a scheduled transfer	Schedule	SYSTEM.FTE.AUTHSCH1. <i>source_agent_name</i>	BROWSE	READ
Delete a scheduled transfer created by the same user				
Delete a scheduled transfer created by any user or group	Schedule operations	SYSTEM.FTE.AUTHOPS1. <i>agent_name</i>	PUT	UPDATE
Cancel a transfer created either by the same user or group that started the transfer, or by another user or group	Transfer operations	SYSTEM.FTE.AUTHOPS1. <i>source_agent_name</i> SYSTEM.FTE.AUTHOPS1. <i>destination_agent_name</i>	BROWSE	READ

Note: To give a user or group permission to set up a resource monitor or scheduled transfer that starts a transfer the user needs both the `Monitor` or `Schedule` authority and `Transfer source` and `Transfer destination` authorities.

V 9.0.0.3 **V 9.0.4** From IBM MQ 9.0.0 Fix Pack 3 and IBM MQ 9.0.4, the order in which authority checks are performed when an agent receives a request to cancel a file transfer is changed so that the agent first checks whether the user requesting the cancellation is the same user who started the transfer. If the user canceling the transfer is not the same as the user that requested it, the agent then checks if the canceling user has `browse` permission on `SYSTEM.FTE.AUTHOPS1.agent_name` queues. The change to the order of checking avoids unexpected errors in agent and queue manager error logs when the user who requested the file transfer and the user who requested the cancellation are the same.

A user can start one agent and want it to interact with another agent. How the two agents can interact depends on the level of access authority that the user has on the other agent authority queue.

Table 32. The level of IBM MQ access authority that the user that starts an agent requires on another agent authority queue so that files can be transferred between the agents.

Agent action	Managed File Transfer access authority	Authority queues	IBM MQ access authority (Multiplatforms)	RACF access level (z/OS only)
Receive a transfer from <i>source_agent</i>	Agent source	SYSTEM.FTE.AUTHAGT1. <i>source_agent_name</i>	BROWSE	READ
Send a transfer to <i>destination_agent</i>	Agent destination	SYSTEM.FTE.AUTHAGT1. <i>destination_agent_name</i>	PUT	UPDATE

Configuring user authority management

To authorize a user to be able to perform an action on an agent, grant the user the appropriate authority on the relevant authority queue. To grant authorities to a user, complete the following steps:

1. Create a user on the system where the agent queue manager is located that has the same name as the user you want to give authority to perform agent actions. This user does not have to be active.
2. Grant the user the appropriate authority on the relevant authority queue. If you are using Linux, UNIX, or Windows, you can use the `setmqaut` command.
3. Refresh the security configuration of the queue manager. You can use the `REFRESH SECURITY MQSC` command.

Example

z/OS The `setmqaut` command is not used on z/OS systems. For z/OS, instead use RACF. See [Setting up security on z/OS](#) for more information.

IBM i The `setmqaut` command is not used on IBM i systems. For IBM i, see [Access authorities for IBM MQ objects](#), which describes how authorization for IBM MQ objects is done. There are three relevant CL commands available on IBM i: **Grant MQ Object Authority (GRTMQMAUT)**, **Revoke MQ Object Authority (RVKMQMAUT)**, and **Refresh MQ Authority (RFRMQMAUT)**.

A user, who is a member of the group `requestor_group`, wants to set up a resource monitor on AGENT1 that transfers a file from AGENT1, which is running under the user `user1`, who is a member of the group `user1_group`, to AGENT2, which is running under the user `user2`, who is a member of the group `user2_group`. AGENT1 connects to QM1; AGENT2 connects to QM2. Both agents have authority checking enabled. To make this possible take the following steps:

1. `requestor` must have **Monitor** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

```
setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHMON1.AGENT1 -g requestor_group +browse
```

2. `requestor` must have **Transfer source** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

```
setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHTRN1.AGENT1 -g requestor_group +browse
```

3. `requestor` must have **Transfer destination** authority on AGENT2. Set this authority by running the following command On the system where QM2 is running:

```
setmqaut -m QM2 -t queue -n SYSTEM.FTE.AUTHTRN1.AGENT2 -g requestor_group +put
```

4. user2 must have **Agent source** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

```
setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHAGT1.AGENT1 -g user2_group +browse
```

5. user1 must have **Agent destination** authority on AGENT2. Set this authority by running the following command on the system where QM2 is running:

```
setmqaut -m QM2 -t queue -n SYSTEM.FTE.AUTHAGT1.AGENT2 -g user1_group +put
```

Logging

If user authority checking is enabled, failed authority checks cause a not authorized log message to be published to the coordination queue manager. See [“MFT message formats for security” on page 783](#) for more information.

Messages about user authority can be written to the agent event log. You can configure the amount of information written to the agent event log by setting the `logAuthorityChecks` property in the agent property file. By default the level of authority check logging is `None`. You can also set the value of `logAuthorityChecks` to `Failures`, which specifies that only failed authorization checks are reported, or `All` which specifies that failed and successful authorization checks are reported.

For more information, see [“The MFT agent.properties file” on page 526](#).

Related reference

[“Restricting group authorities for MFT-specific resources” on page 337](#)

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

[“Authorities for the MFT logger” on page 345](#)

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

[“fteStopAgent” on page 508](#)

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

Authorities for the MFT logger

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

The operating system user who runs the logger requires the following IBM MQ authorities:

- CONNECT and INQUIRE on the coordination queue manager.
- SUBSCRIBE permission on the SYSTEM.FTE topic.
- PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
- GET permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

Related reference

[“Restricting group authorities for MFT-specific resources” on page 337](#)

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

[“Restricting user authorities on MFT agent actions” on page 342](#)

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

MFT permissions to access sensitive configuration information

Any file used to store sensitive configuration information, meaning any file referenced from the IBM MQ configuration tree, must not have system-wide read, write, or (where applicable), delete permissions. These restrictions also apply to truststore and keystore files.

If a Managed File Transfer process detects a condition that a configuration file contains sensitive information, is a keystore or truststore file, and has system-wide read, write, or delete permissions, the process takes one of the following actions:

- Fails to start, if the condition is detected at startup time.
- Generates a warning message and ignores the contents of the configuration file if the condition was detected at runtime. This is relevant to the protocol bridge and the Connect:Direct bridge, which reload a configuration if it changes while the process is running.

On systems with a UNIX type file system

The criteria for determining that a file has unacceptable system-wide permissions are:

- The others class has been granted read permission on the file
- The others class has been granted write permission on the file
- The others class has been granted write permission on the directory containing the file

On Windows systems

The criteria for determining that a file has unacceptable system-wide permissions are:

- Any of the Everyone, Guests, or Users groups have any of the following permissions:
 - Read data permission on the file
 - Append data permission on the file
 - Write data permission on the file
- Any of the Everyone, Guests, or Users groups has Create files permission on the folder containing the file and they also have any of the following permissions:
 - Delete subfolders and files permission on the folder containing the file
 - Delete permission on the file

Authority to publish MFT Agents log and status messages

Managed File Transfer Agents issue various log, progress, and status messages that are published on the coordination queue manager. The publication of these messages is subject to the IBM MQ security model, and in some cases you might have to perform further configuration to enable publication.

For more information about IBM MQ security, see the section starting with [Securing](#).

Managed File Transfer agents flow messages for publication to the SYSTEM.FTE queue on the coordination queue manager. Each message carries a user ID in its message descriptor (MQMD). Messages are published using a topic object that is also called SYSTEM.FTE. For the publication of a given message to take place, the authority records of the SYSTEM.FTE topic must permit publication by the user ID contained in the MQMD of the message.

On z/OS, the channel initiator user ID needs access to publish to the SYSTEM.FTE topic. The user ID in the MQMD of the message also needs access to publish to this topic if the [RESLEVEL security profile](#) causes two user IDs to be checked for the channel initiator connection.

The user ID initially contained in the message depends on how the agent is connected to its own queue manager. Messages from bindings-connected agents contain the user ID that the agent is running under. Messages from client-connected agents contain an internal IBM MQ user ID.

You can change the user ID in a message. For both client- and bindings-connected agents, you can use the property `publicationMDUser` (in the `agent.properties` file) to specify a user ID, which is used in all log and status messages from that agent. The agent must be given permission by its own queue manager to use this alternative user ID; give this permission by granting `setid` authority to the user ID that the agent runs under.

You can also change the user ID contained in all messages from a client-connected agent using the `MCAUSER` property on the channel that the agent uses to connect to its queue manager.

You can change the user ID in messages using a channel exit, for example on the receiver channel bringing messages into the coordination queue manager.

Depending on the IBM MQ topology and policies, there are a number of ways an IBM MQ administrator can use the information in this topic to ensure that the publication of status and log messages takes place. Two examples are:

- Determine all the user IDs used by agents in the network. Explicitly grant an authority record for each of these IDs.
- Create one or more common user names to publish log and status messages. Create authority records for these user names on the coordination queue manager. Set the `publicationMDUser` property for each agent to a common user name. On each agent queue manager, grant `setid` authority to the user ID that the agent runs under to allow it to accept the `publicationMDUser` property.

Authorities for MFT to access file systems

For any file transfer request, the Managed File Transfer agent processes require some level of access to their local file systems.

- To transfer from a source file, the user ID that the source agent runs under must have read access to the source file. Additionally, you might need to give the source agent delete or write authority depending on the source disposition attribute.
- To transfer to a file or directory, the user ID that the destination agent runs under must have write authority to the specified path. Additionally, you might need to give the destination agent update authority, depending on the destination exists attribute.
- In addition to the file access authority that you grant to the agent process, you can also use sandboxing to specify and enforce a restricted file path area. For more information, see [“MFT sandboxes” on page 45](#).
- If the files that you want to transfer to or from are not in a location accessible to the agent, for example  a VSAM data set or in a location that is restricted by the sandboxing capability, you can use Managed File Transfer user exits to move the file to or from a location that can be accessed by the agent. For more information, see [“Customizing MFT with user exits” on page 245](#).

commandPath MFT property

Use the `commandPath` property to specify the locations that Managed File Transfer can run commands from. Take extreme care when you set this property because any command in one of the specified `commandPaths` can effectively be called from a remote client system that is able to send commands to the agent.

You can specify a command to be run on the system where the agent is running from the managed transfer and managed call functions of Managed File Transfer. See [Program invocation](#) for information. However, commands must be on paths referenced by the `commandPath` agent property.

If the command specified is not fully qualified, Managed File Transfer attempts to find a matching command on the command path. If there is more than one matching command on the command path, the first match is used.

By default, the `commandPath` property is empty so that the agent cannot call any commands.

Specify the `commandPath` agent property as follows:

```
commandPath=command_directory_name
separator...command_directory_name
```

z/OS Or for z/OS only, specify:

```
commandPath=command_directory_name_or_data_set_name_prefix
separator...command_directory_name_or_data_set_name_prefix
```

where:

- `command_directory_name` is a directory path for commands that can be run.
- **z/OS** `command_directory_name_or_data_set_name_prefix` is a z/OS UNIX System Services directory path for commands that can be run, or a data set name prefix, that starts with `//`. You can choose to use a fully qualified or unqualified data set name prefix (that is, in the form: `// 'HLQ. . . '` or `//HLQ. . .`). Specify partitioned data sets in the form `// 'HLQ() . . . '` or `//HLQ() . . .`. Use data sets to specify JCL script commands only.
- `separator` is the platform-specific separator.

UNIX For example, on a UNIX system if you want to run commands that are located in the directories `/home/user1/cmds1` and `/home/user1/cmds2`, set the `commandPath` agent property as follows:

```
commandPath=/home/user1/cmds1:/home/user1/cmds2
```

Windows For example, on a Windows system if you want to run commands that are located in the directories `C:\File Transfer\commands` and `C:\File Transfer\agent commands`, set the `commandPath` agent property as follows:

```
commandPath=C:\\File Transfer\\commands;C:\\File Transfer\\agent commands
```

On a Windows system the separator character, backslash (`\`), must be escaped and be entered as a double backslash (`\\`). The backslash character (`\`) can also be replaced with a forward slash (`/`).

z/OS For example, on z/OS if you want to run commands that are:

- In the directories `/home/user1/cmds1` and `/home/user1/cmds2`
- In data sets that start with `// 'USER.CMD1 '`, `//CMD2`,
- Members of a fully qualified PDS named `// 'USER.CMDS '`

set the `commandPath` agent property as follows:

```
commandPath=/home/user1/cmds1:/home/user1/cmds2:// 'USER.CMD1 ' ://CMD2:// 'USER.CMDS() '
```

Important: Extreme care must be taken when you set this property, because any command in one of the specified `commandPaths` can be called from a remote client system that is able to send commands to the agent. For this reason, by default, when you specify a `commandPath`, sandboxing is configured so that all `commandPath` directories (and their subdirectories) are automatically denied access for a transfer:

- If the agent is configured to use an agent sandbox, the `commandPath` directories are automatically added to the list of denied directories when the agent starts.
- If the agent is configured with one or more user sandboxes, the `commandPath` directories are added as `<exclude>` elements to the `<read>` and `<write>` elements for each user sandbox when the agent starts up.

- If the agent is not configured to use either an agent sandbox, or user sandboxes, then a new agent sandbox is created when the agent starts up that has the commandPath directories specified as denied directories.

V9.0.0.1 You can override this behavior for compatibility with the following releases:

- IBM WebSphere MQ File Transfer Edition.
- The IBM WebSphere MQ 7.5.0 Fix Pack 1 Managed File Transfer component (or earlier).
- The IBM WebSphere MQ 7.5.0 Fix Pack 2 Managed File Transfer component (or later) on an installation that does not have the installation property enableFunctionalFixPack=7502 set.

You can override this behavior by adding the following property to the agent.properties file:

```
addCommandPathToSandbox=false
```

V9.0.0.1 When the addCommandPathToSandbox property is present and set to false, the following behavior occurs:

- If the agent is configured to use an agent sandbox, and the sandbox does not have any allowed directories specified, the commandPath directories are automatically added to the list of denied directories when the agent starts.
- If the agent is configured to use an agent sandbox, and the sandbox has one or more allowed directories specified, the commandPath directories are not added to the list of denied directories when the agent starts.
- If the agent is configured with one or more user sandboxes, the user sandboxes are not changed, and the commandPath directories are not added as <exclude> elements to the <read> and <write> elements for each user sandbox.
- If the agent is not configured to use either an agent sandbox, or user sandboxes, then a new agent sandbox is created when the agent starts up that has the commandPath directories specified as denied directories.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, agent.properties, that must contain the information that an agent uses to connect to its queue manager. The agent.properties file can also contain properties that alter the behavior of the agent.

Which MFT commands and processes connect to which queue manager

A Managed File Transfer topology consists of a number of different components.

These components are:

- One or more agents, with their associated agent queue manager
- A coordination queue manager
- A command queue manager
- A number of commands that are used to administer the topology, and submit managed transfers
- An optional logger, which collects information about the managed transfers that are performed by the agents in the topology
- The IBM MQ Explorer Managed File Transfer plugin, which can be used to perform some administrative tasks and view information about managed transfers.

Agents, loggers, commands, and the IBM MQ Explorer Managed File Transfer plugin connect to one or more queue managers when they run.

The following tables summarizes which queue manager agents, loggers, commands, and IBM MQ Explorer Managed File Transfer plugin connect to, when they run.

If there are no X characters for a command or process in the table, the command does not connect to any queue manager or process when it is run.

Table 33. Summary of which Managed File Transfer commands connect to which queue manager

Command name	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
fteAnt				
fteCancelTransfer		X		
fteChangeDefaultConfigurationOptions				
fteCleanAgent	X			
fteCreateAgent	X			
fteCreateBridgeAgent	X			
fteCreateCDAgent	X			
 fteCreateEnvironment				
fteCreateLogger				
fteCreateMonitor		X		
fteCreateTemplate			X	
fteCreateTransfer		X		
fteDefine				
fteDelete				
fteDeleteAgent	X		X	
fteDeleteLogger				
fteDeleteMonitor		X		
fteDeleteScheduledTransfer		X		
fteDeleteTemplates			X	
fteDisplayVersion				
fteListAgents			X	
fteListMonitors			X	
fteListScheduledTransfers			X	
fteListTemplates			X	
fteMigrateAgent				
fteMigrateConfigurationOptions				
fteMigrateLogger				
fteModifyAgent				
fteModifyLogger				
fteObfuscate				
ftePingAgent		X		
fteRAS				
 fteSetAgentLogLevel				
fteSetAgentTraceLevel				
fteSetLoggerTraceLevel				

Table 33. Summary of which Managed File Transfer commands connect to which queue manager (continued)

Command name	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
 fteSetProductID				
fteSetupCommands				
fteSetupCoordination				
fteShowAgentDetails			X	
fteShowLoggerDetails				
fteStartAgent				
fteStartLogger				
fteStopAgent		X		
fteStopLogger		X		

Table 34. Summary of which Managed File Transfer processes connect to which queue manager

Processes	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
Managed File Transfer agents	X			
Managed File Transfer plug-in for IBM MQ Explorer		X	X	
Managed File Transfer logger			X	X

The file that contains the credentials information that is required to connect to each type of queue manager, that is, agent, command, and coordination queue managers, can be specified in the associated properties file. For example, the coordination queue manager has a `coordination.properties` file. In this file, you can set the **coordinationQMGrAuthenticationCredentialsFile** property to point to the credentials file.

The commands that connect to the coordination queue manager use the credentials information that is specified in that file. If security is enabled on a queue manager and this property is incorrectly set, MFT commands do not successfully complete. For more information, see [MFT and IBM MQ connection authentication](#).

Related concepts

[Installed MFT command sets](#)

Details of which MFT commands connect to which queue manager

Further information on which Managed File Transfer commands connect to which queue manager.

This topic expands on the information in [“Which MFT commands and processes connect to which queue manager”](#) on page 349, together with some illustrations.

Commands that connect to the coordination queue manager

The following commands connect to the coordination queue manager:

- [fteCreateTemplate](#)
- [fteDeleteAgent](#)
- [fteDeleteTemplates](#)
- [fteListAgents](#)
- [fteListMonitors](#)

- [fteListScheduledTransfers](#)
- [fteListTemplates](#)
- [fteShowAgentDetails](#)

The coordination queue manager for a Managed File Transfer topology is a central hub that has knowledge of the entire topology. The coordination queue manager is connected to all of the agent queue managers in a topology through sender and receiver channels. Agents regularly publish status information to the coordination queue manager, and store their transfer templates there.

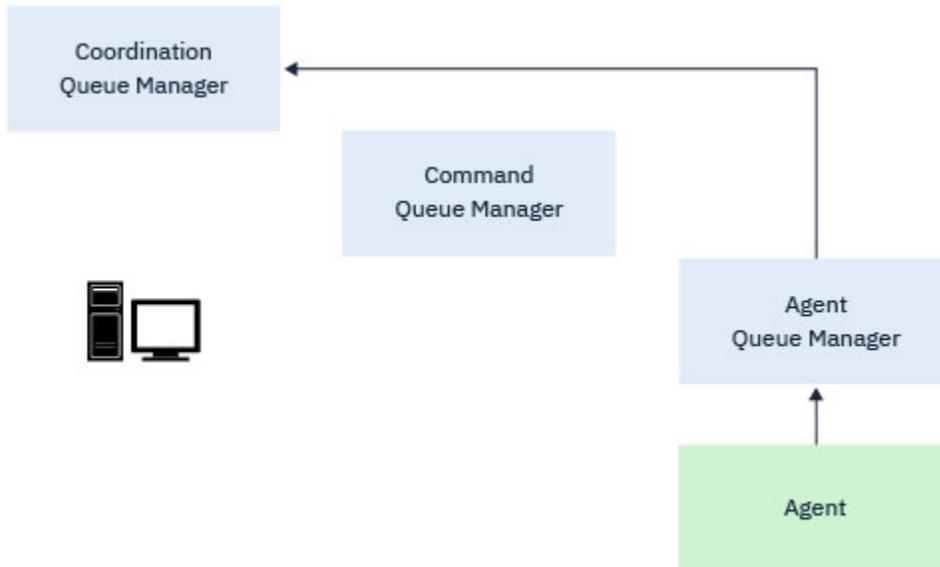


Figure 9. Agents publish status information or store transfer templates on the coordination queue manager

When any of the preceding commands that connect to the coordination queue manager are run, they connect directly to the coordination queue manager and either:

- Create or delete a transfer template.
- Query state information about agents, monitors, or scheduled transfers, and display that information to the user.

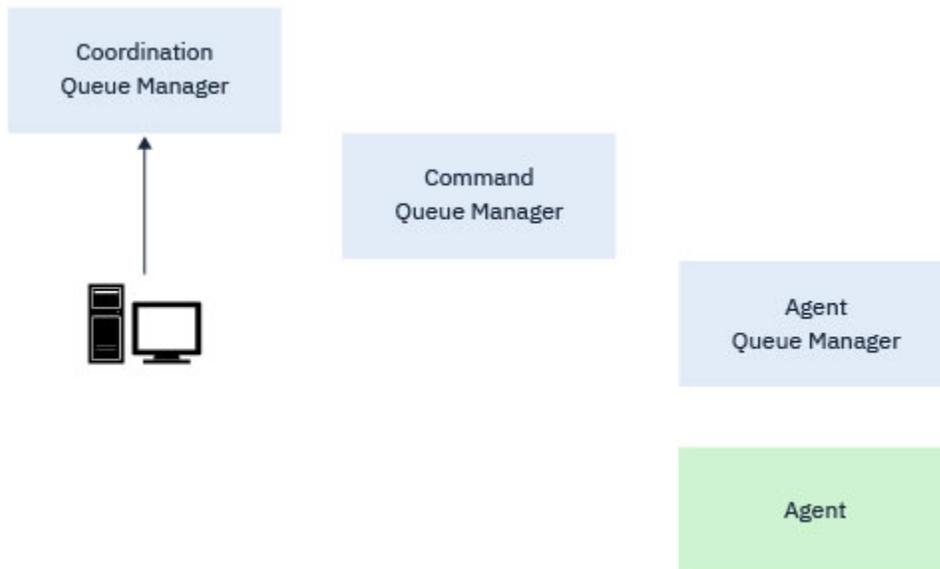


Figure 10. Commands then connect directly to the coordination queue manager to either obtain the appropriate status information or work with transfer templates.

Commands that connect to the command queue manager

The following commands connect to the command queue manager:

- [fteCancelTransfer](#)
- [fteCreateMonitor](#)
- [fteCreateTransfer](#)
- [fteDeleteMonitor](#)
- [fteDeleteScheduledTransfer](#)
- [ftePingAgent](#)
- [fteStopAgent](#)

You can think of the command queue manager as a form of gateway into a Managed File Transfer topology. It is connected to agent queue managers using sender and receiver channels.

When any of the preceding commands that connect to the queue manager are run, they:

- Connect to the command queue manager.
- Create a temporary reply queue.
- Send a message containing the command details to the appropriate agent.

The message is routed through the IBM MQ network to the agent queue manager, where it is picked up by the agent and processed.

After the agent has processed the command, the agent sends a reply back to the command queue manager, where the reply is picked up by the command.

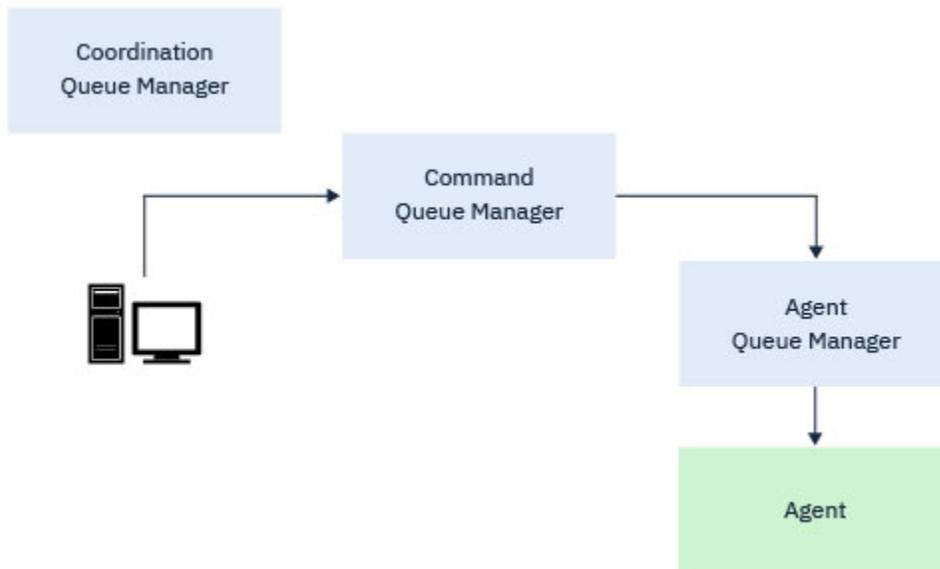


Figure 11. The commands connect to the command queue manager. The message containing the command is then routed through the IBM MQ network to the correct agent queue manager, where it is picked up by the agent.

Commands that connect to the agent queue manager

The following commands connect to the agent queue manager:

- [fteCleanAgent](#)
- [fteCreateAgent](#)
- [fteCreateBridgeAgent](#)
- [fteCreateCDAgent](#)
- [fteDeleteAgent](#)

Every agent has its own agent queue manager. The agent uses system queues hosted on this queue manager to maintain state information and receive requests for work.

A single queue manager can act as the agent queue manager for multiple agents. Agent queue managers are connected to the coordination queue manager, the command queue manager, and other agent queue managers using sender and receiver channels.

The **fteCreateAgent**, **fteCreateBridgeAgent**, and **fteCreateCDAgent** commands take the agent queue manager name as an argument.

When these commands are run, they connect to the specified queue manager and send a message to the coordination queue manager indicating that the agent has been added to the Managed File Transfer topology.

Similarly, when **fteDeleteAgent** is run, it connects to the agent queue manager and sends a message to the coordination queue manager, informing it that the agent has now been removed from the Managed File Transfer topology.

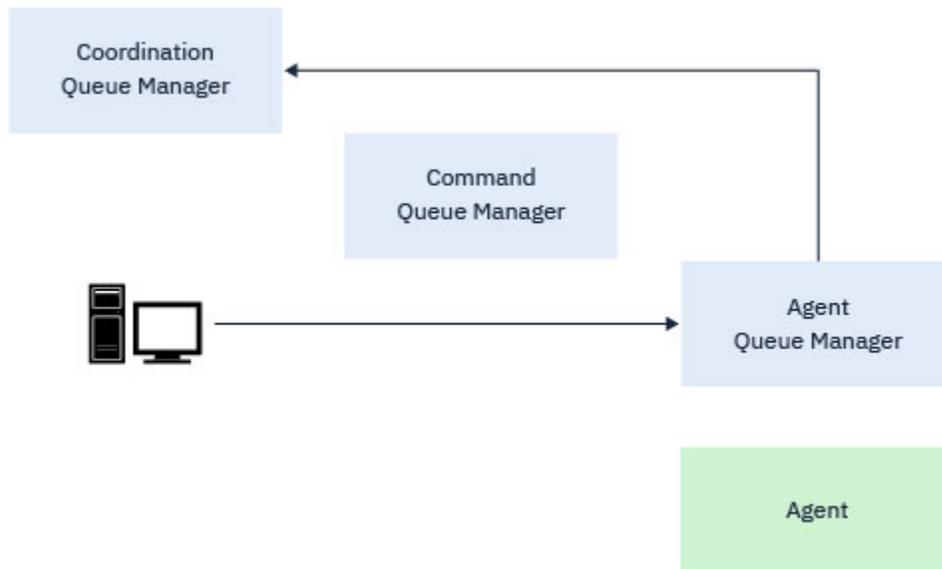


Figure 12. The various **fteCreate** commands, along with the **fteDeleteAgent** command, connect to the agent queue manager and send a message to the coordination queue manager to either register or deregister an agent from the topology.



Attention: fteCleanAgent connects to the agent queue manager, and removes any state information for that agent from its system queues.

Running this command could have an impact on the whole topology. As such, you should only run this command under guidance from IBM.

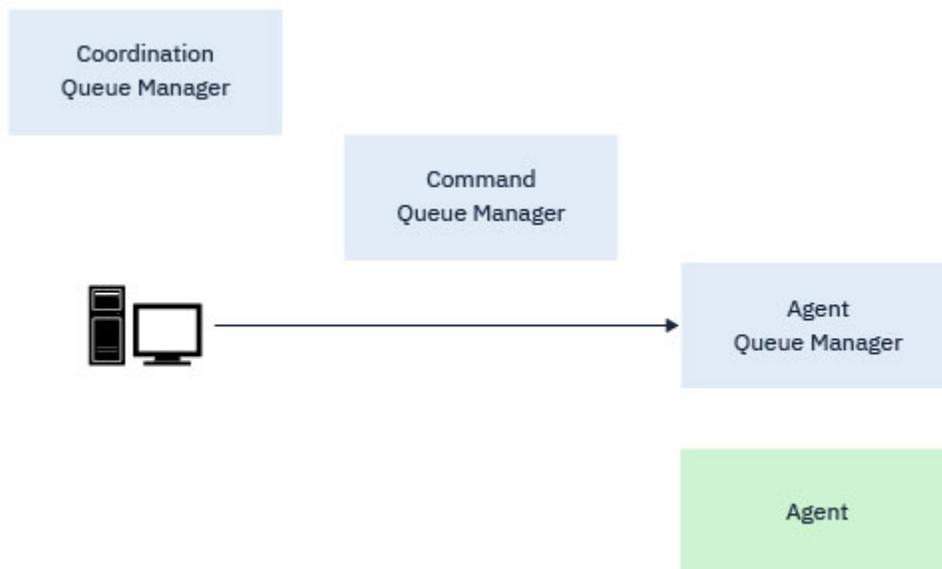


Figure 13. The **fteCleanAgent** command connects directly to the agent queue manager, and removes state information from the system queues of the agent

Related concepts

[Installed MFT command sets](#)

MFT commands

All Managed File Transfer commands are listed with links to their detailed descriptions.

<i>Table 35. Managed File Transfer commands and their purpose</i>	
Command name	Purpose
Commands for migration:	
fteMigrateAgent	Migrate an agent and its configuration from IBM WebSphere MQ File Transfer Edition 7.0.4 to Managed File Transfer 7.5 or later
fteMigrateConfigurationOptions	Migrate an IBM WebSphere MQ File Transfer Edition 7.0 configuration to Managed File Transfer in IBM WebSphere MQ 7.5 or later.
fteMigrateLogger	Migrate the configuration of a stand-alone database logger from IBM WebSphere MQ File Transfer Edition 7.0.1 or later to Managed File Transfer 7.5 or later.
Commands for configuration:	
fteChangeDefaultConfigurationOptions	Change the default configuration options that you want Managed File Transfer to use
fteCreateAgent	Create a Managed File Transfer Agent
fteCreateBridgeAgent	Create a Managed File Transfer protocol bridge agent
fteCreateCDAgent	Create a Managed File Transfer Connect:Direct bridge agent
V 9.0.1 fteCreateEnvironment	Set the environment variable for the configuration and transfer of files for the Redistributable Managed File Transfer Agent.
fteCreateLogger	Create a Managed File Transfer logger
fteDefine	Generate the configuration scripts necessary to define the specified objects.
fteDelete	Generate the configuration scripts necessary to remove the specified objects.
fteDeleteAgent	Delete a particular Managed File Transfer Agent
fteDeleteLogger	Delete a Managed File Transfer logger
fteModifyAgent	Windows only. Modify an agent, Connect:Direct bridge agent, or protocol bridge agent to run as a Windows service.
fteModifyLogger	Windows only. Modify the logger to run as a Windows service.
fteSetupCommands	Specify the details of the queue manager that connects to the IBM MQ network when you issue commands
fteSetupCoordination	Configure a Managed File Transfer coordination queue manager
Commands for administration:	
fteAnt	Run an Ant script in an environment with file transfer Ant tasks available.
fteCancelTransfer	Cancel a file transfer
fteCleanAgent	Clean up the queues used by an agent
fteCreateMonitor	Create and start a new resource monitor
fteCreateTemplate	Create a transfer template for future use
fteCreateTransfer	Create and start a new file transfer
fteDeleteMonitor	Stop and remove an existing resource monitor
fteDeleteScheduledTransfer	Delete a particular file transfer that you have previously scheduled
fteDeleteTemplates	Delete existing file transfer templates

Table 35. Managed File Transfer commands and their purpose (continued)

Command name	Purpose
fteListAgents	List all of the agents registered against a particular coordination queue manager
fteListMonitors	List all of the resource monitors registered against a particular coordination queue manager
fteListScheduledTransfers	List all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.
fteListTemplates	List all the file transfer templates for a coordination queue manager
ftePingAgent	Pings an agent to determine whether the agent is active and able to process transfers.
  fteSetProductID	Set z/OS SCRT recording product id
  fteShowAgentDetails	Display the details of a particular agent
fteShowLoggerDetails	Display the details of a particular logger
fteStartAgent	Start a particular agent before using it to transfer files
fteStartLogger	Start logger
fteStopAgent	Stop a particular agent
fteStopLogger	Stop logger
Command for security:	
fteObfuscate	Encrypt sensitive data in credentials files.
Commands for troubleshooting:	
fteDisplayVersion	Display the product version
 fteSetAgentLogLevel	Enable or disable diagnostic information logging for file transfers between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers.
fteSetAgentTraceLevel	Set the level of agent trace to run
fteSetLoggerTraceLevel	Set the level of logger trace to run
fteRAS	Run the RAS gathering tool

See [Installed MFT command sets](#) for a table showing which commands are installed with which Managed File Transfer offering.

The syntax for each command and its parameters is presented in the form of a syntax diagram called a railroad diagram. For information about how to interpret railroad diagrams, see [How to read railroad diagrams](#).

Authority to use MFT commands

Your user ID must be a member of the mqm group if you want to issue Managed File Transfer commands, unless you have already configured IBM MQ to allow users who are not in the mqm group to issue commands.

 For more information about defining an alternative group to mqm on z/OS, see [Issuing commands to IBM MQ for z/OS](#)

For more information about authorization, see [Authority to administer IBM MQ](#).  If you are using IBM i, start with the following topic: [IBM MQ authorities](#).

A subset of the Managed File Transfer commands can be issued using the IBM MQ Explorer.

Issuing commands from Windows and UNIX systems

Note the following environment-specific information for issuing commands:

Managed File Transfer for Windows

All commands can be issued from a command line. Command names are not case-sensitive: You can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to control commands (such as queue names) and parameters (such as **-m** for queue manager name) are case-sensitive.

In the syntax descriptions, the hyphen (-) is used as a flag indicator.

Managed File Transfer for UNIX systems

All Managed File Transfer commands can be issued from a shell. All commands are case-sensitive.

Issuing commands from z/OS systems



The Managed File Transfer commands are installed in the bin subdirectory of the location chosen when the product was installed. The commands can be run from either of the following options:

- Directly from the USS environment by specifying the path to the command or including the bin subdirectory in the user command path.
- From a PDSE data set of commands configured from the PDSE command template library, for a particular agent or logger. For more information, see [“Creating an MFT Agent or Logger command data set”](#) on page 62.

Issuing commands from the IBM i platform



Note the following environment-specific information for issuing commands on IBM i:

- You can start Managed File Transfer commands using the Qshell interpreter. To start the Qshell interpreter, issue the **STRQSH** command from an IBM i system command line.
- When you run commands in the Qshell environment, command names are not case-sensitive: You can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to control commands (such as queue names) and parameters (such as **-m** for queue manager name) are case-sensitive.

Related reference

[“Return codes for MFT”](#) on page 322

Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

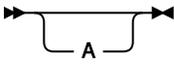
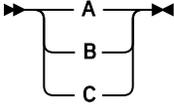
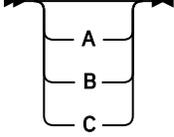
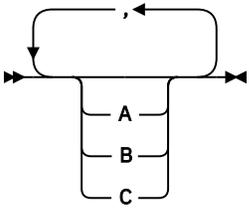
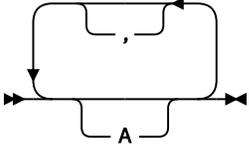
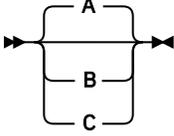
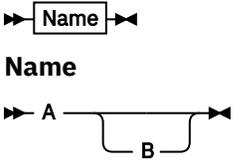
How to read railroad (syntax) diagrams

Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in railroad diagrams are:

<i>Table 36. How to read railroad diagrams</i>	
Convention	Meaning
▶ A — B — C ◀	You must specify values A, B, and C. Required values are shown on the main line of a railroad diagram.

Table 36. How to read railroad diagrams (continued)

Convention	Meaning
	<p>You may specify value A. Optional values are shown below the main line of a railroad diagram.</p>
	<p>Values A, B, and C are alternatives, one of which you must specify.</p>
	<p>Values A, B, and C are alternatives, one of which you might specify.</p>
	<p>You might specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.</p>
	<p>You might specify value A multiple times. The separator in this example is optional.</p>
	<p>Values A, B, and C are alternatives, one of which you might specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.</p>
	<p>The railroad fragment Name is shown separately from the main railroad diagram.</p>
<p>Punctuation and uppercase values</p>	<p>Specify exactly as shown.</p>
<p>Lowercase values (for example, <i>name</i>)</p>	<p>Supply your own text in place of the <i>name</i> variable.</p>

Tracing MFT commands

You can trace any of the Managed File Transfer commands to help with problem determination from the command line.

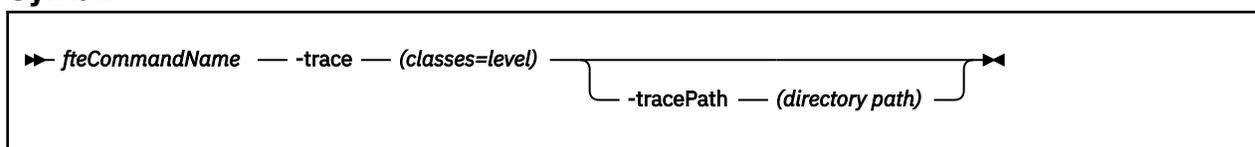
Purpose

Use the **-trace** parameter for any MFT command to enable trace at a specified level. The trace files produced are located in your current working directory unless the **-tracePath** parameter is included to identify a different directory.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the agent .properties file. These properties are described in [Advanced agent properties](#).

Syntax



Parameters

-trace (classes=level)

Required. Level to set the trace and which classes to apply the trace to. Specify the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

which traces all Managed File Transfer classes.

Specify a colon-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off

Switches the agent trace off but continues to write information to the log files. This is the default option.

flow

Captures data for trace points associated with processing flow in the agent.

moderate

Captures a moderate amount of diagnostic information in the trace.

verbose

Captures a verbose amount of diagnostic information in the trace.

all

Sets agent trace to run on all agent classes.

-tracePath (directory path)

Optional. Specify the directory that you want the trace to be written to. For example, `c:\temp`.

z/OS If you do not specify this parameter, the value is the directory that the command was issued from. For example, on z/OS:

z/OS `/u/smith/fte/wmqmft/mqft/logs/MQPV/loggers/BFGLG1/logs/`

This parameter is valid only when the **-trace** parameter is specified.

Example

In this example the trace level is set to all, meaning that all of the classes belonging to AGENT.NAME are traced for the **fteStartAgent** command:

Note: When the agent is started, the trace goes to `mft_config/logscoordination_qmgr/agents/agent`

```
fteStartAgent -trace com.ibm.wmqfte=all -tracePath /u/mft/trace AGENT.NAME
```

In this example the trace level is set to moderate for the `com.ibm.wmqfte.common` classes for the agent AGENT.NAME. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate AGENT.NAME
```

In this example the trace level is set to moderate for the `com.ibm.wmqfte.common` classes for the agent AGENT.NAME, and the trace is written to the `c:\$user` directory. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate -tracePath c:\$user AGENT.NAME
```

Related reference

[“Troubleshooting MFT” on page 259](#)

Use the following reference information to help you to diagnose errors in Managed File Transfer:

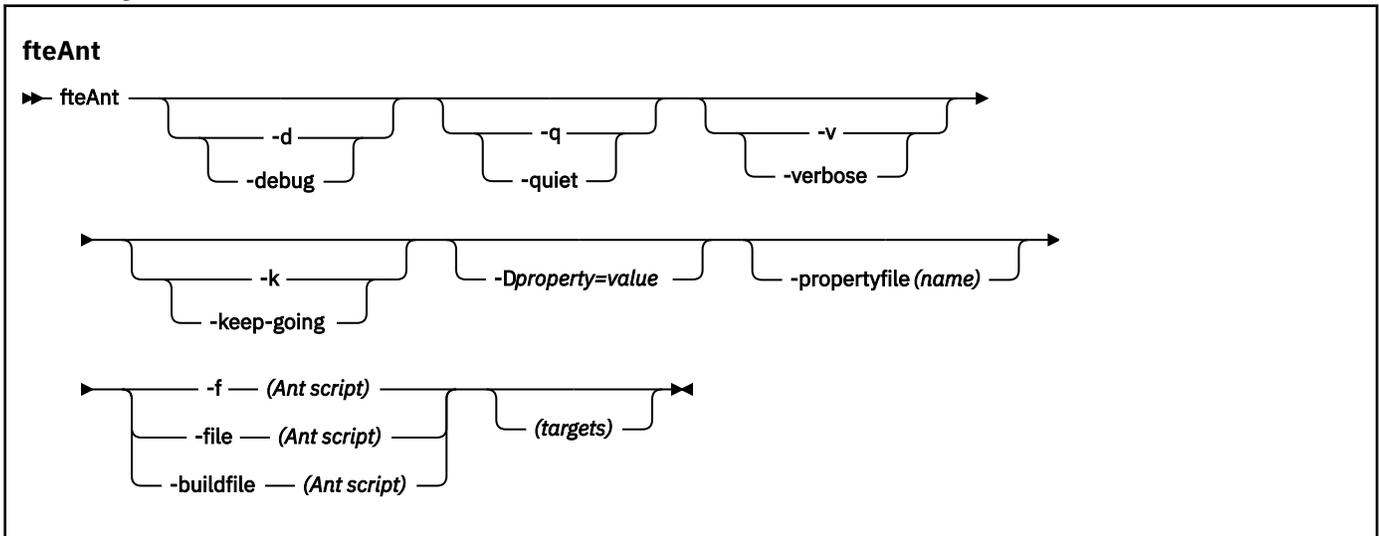
fteAnt: run Ant tasks in MFT

The **fteAnt** command runs Ant scripts in an environment that has Managed File Transfer Ant tasks available.

Purpose

Use the **fteAnt** command to run an Ant script in an environment with Managed File Transfer. Unlike the standard **ant** command, **fteAnt** requires that you define a script file.

Syntax



Parameters

-debug or -d

Optional. Generate debugging output.

-quiet or -q

Optional. Generate minimal output.

-verbose or -v

Optional. Generate verbose output.

-keep-going or -k

Optional. Execute all targets that do not depend on failed targets.

-D property=value

Optional. Use *value* for a given *property*. Properties that are set with **-D** take precedence over those set in a properties file.

Use the property **com.ibm.wmqfte.propertyset** to specify the set of configuration options that are used for Ant tasks. Use the name of a non-default coordination queue manager as the value for this property. Ant tasks then use the set of configuration options that are associated with this non-default coordination queue manager. If you do not specify this property, the default set of configuration options that are based on the default coordination queue manager is used. If you specify the **cmdqm** attribute for an Ant task, this attribute takes precedence over the set of configuration options that are specified for the **fteAnt** command. This behavior applies regardless of whether you are using the default set of configuration options or specifying a set with the **com.ibm.wmqfte.propertyset** property.

-propertyfile (name)

Optional. Load all properties from a file with **-D** properties taking precedence.

-f (Ant script), -file (Ant script), or -buildfile (Ant script)

Required. Specifies the name of the Ant script to run.

targets

Optional. The name of one or more targets to run from the Ant script. If you do not specify a value for this parameter, the default target for the script is run.

-version

Optional. Displays the Managed File Transfer command and Ant versions.

-? or -h

Optional. Displays command syntax.

Example

In this example, the target **copy** in Ant script `fte_script.xml` is run and the command writes debugging output to standard out.

```
fteAnt -d -f fte_script.xml copy
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Other status return codes can also be specified from Ant scripts, for example by using the Ant fail task.

See [Fail](#) for more information.

z/OS fteBatch, fteCommon and ftePlatform

The **z/OS** `fteBatch`, `fteCommon` and `ftePlatform` are scripts that are provided by Managed File Transfer in the `MQ_INSTALLATION_PATH/bin` directory as helper scripts. Not all of these scripts are present on every platform.

fteBatch script (z/OS only)**z/OS**

`fteBatch` is a helper script for running Managed File Transfer from the JZOS Batch Launcher. `fteBatch` is installed on z/OS only. Typically Managed File Transfer is started using the supplied command shell scripts, which perform some environment configuration before starting the Java class appropriate to that function. When Managed File Transfer is started using the JZOS Batch Launcher, the Java class is started directly from the Launcher. `fteBatch` can be called as part of the launcher setup to place the required class name into an environment variable and performs the setup work that the normal command shell scripts perform before starting Java. This provides a level of isolation between your jobs and the internal class names used by Managed File Transfer.

The `fteBatch` command is deprecated for Managed File Transfer 8.0, as you can run Managed File Transfer through the new PDSE data set of commands. For more information, see [“Creating an MFT Agent or Logger command data set”](#) on page 62.

fteCommon

`fteCommon` is a helper script started by the other Managed File Transfer command scripts to perform common setup processing before starting Java.

ftePlatform

`ftePlatform` is a helper script started by the `fteCommon` script to perform platform-specific setup processing.

fteCancelTransfer

Use the **fteCancelTransfer** command to cancel a Managed File Transfer transfer. You can issue this command against either the source or destination agent for the transfer.

Purpose

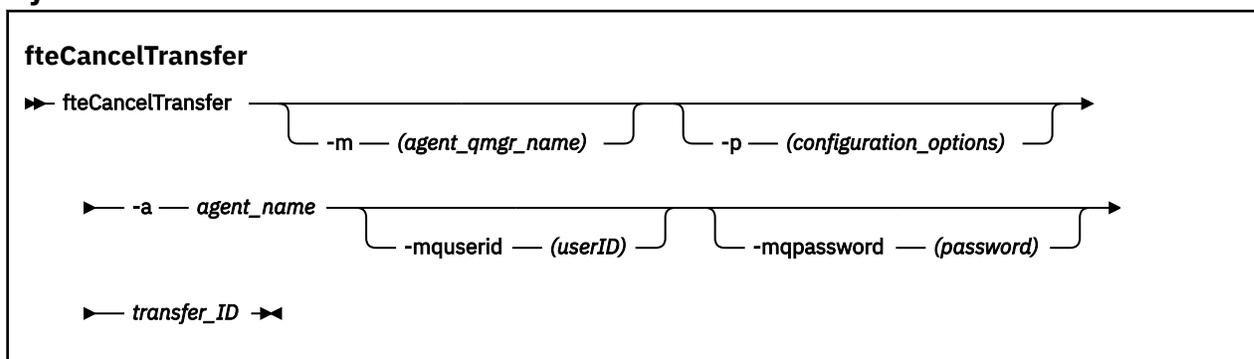
If you issue the **fteCancelTransfer** command while that transfer is currently in progress, any files already transferred as part of that transfer remain on the destination system and are not deleted. Any files partially transferred as part of that transfer are deleted from the destination system. The destination side of the transfer logs that transfer as "cancelled".

If a transfer to a Connect:Direct node is canceled, any files partially transferred as part of the canceled transfer remain on the destination system and are not deleted.

You can run the **fteCancelTransfer** command from any system that can connect to the IBM MQ network and then route to the agent queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured Managed File Transfer on this system to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Syntax



Parameters

-m (agent_qmgr_name)

Optional. The name of the agent queue manager. This agent must be either the source or destination agent for the transfer you want to cancel. If you do not specify this parameter, the cancel request is sent to the queue manager identified by the set of configuration options you are using.

-p (configuration_options)

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-a (agent_name)

Required. The name of either the source or destination agent of the transfer that you want to cancel.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

transfer_ID

Required. The ID of the transfer you want to cancel. The transfer ID (also known as the request ID) is displayed at the command line after you issue the **fteCreateTransfer** command. Transfer IDs are also included in file transfer log messages or are displayed in the IBM MQ Explorer Transfer Log panel.

-? or -h

Optional. Displays command syntax.

Example

In this example AGENT1 is the source agent for the transfer to be canceled.

```
fteCancelTransfer -a AGENT1 414d5120514d5f4c4d343336303920201159c54820027102
```

Return codes**0**

Either the command completed successfully or the specified transfer ID is unknown to the agent. If the transfer ID is unknown to the agent, the most likely reason is that the transfer has already completed or has been canceled.

1

Command ended unsuccessfully.

Related reference

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

fteChangeDefaultConfigurationOptions

Use the **fteChangeDefaultConfigurationOptions** command to change the default configuration options that you want Managed File Transfer to use. The value of the configuration options defines the group of properties files that Managed File Transfer uses.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Purpose

Your default Managed File Transfer configuration options are established the first time you use the `fteSetupCoordination` command to configure a queue manager as the coordination queue manager. During the installation of the MFT product, the `mqft` directory is created under `<MQ_DATA_PATH>` if it does not already exist. Additionally, configuration, installations, and logs directories are created under the `mqft` directory, if they do not already exist.

By using the `fteChangeDefaultConfigurationOptions` command you can change the default coordination queue manager that is defined in the `installation.properties` file. If you change this coordination queue manager, Managed File Transfer uses the configuration options given by the structured set of directories and property files contained in the directory you used as input for `configuration_options` by default. This directory name is the same as the coordination queue manager used by agents under this configuration.

See “MFT configuration options on Multiplatforms” on page 56 for more information about the `installation.properties` file.

Syntax

`fteChangeDefaultConfigurationOptions`

```
► fteChangeDefaultConfigurationOptions — configuration_options ◄
```

Parameters

`configuration_options`

Required. This parameter specifies the default configuration options that you want to change to. Use the name of a non-default coordination queue manager as the input for this parameter.

`-? or -h`

Optional. Displays command syntax.

Example

In this example, the default configuration options are changed to `QM_COORD2`:

```
fteChangeDefaultConfigurationOptions QM_COORD2
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

“MFT configuration options on Multiplatforms” on page 56

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

`fteCleanAgent`: clean up an MFT Agent

Use the `fteCleanAgent` command to clean up the queues that a Managed File Transfer Agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the

fteCleanAgent command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

Purpose

Use the **fteCleanAgent** command to delete messages from the persistent and non-persistent queues used by the agent. Specifically, this command can carry out the following actions:

- Remove any transfers that were in progress to this agent or from this agent before the transfer was stopped. These transfers are not resumed when the agent restarts
- Remove any commands that have already been submitted to the agent, but have not yet been carried out
- Delete all resource monitors stored on the agent
- Delete all scheduled transfers stored on the agent
- Delete all invalid messages stored on the agent

If the agent is a Connect:Direct bridge agent, the **-ms**, **-ss**, and **-ims** parameters are not valid. For Connect:Direct bridge agents the command also carries out the following actions:

- Deletes all files from the directory where the Connect:Direct bridge agent temporarily stores files while they are being transferred. The location of this directory is defined by the **cdTmpDir** parameter
- Displays information about the Connect:Direct processes that are associated with any ongoing transfers

Before IBM MQ 9.0.3 and IBM MQ 9.0.0 Fix Pack 2, if you run the **fteCleanAgent** command with just the **agent_name** parameter, by default the command runs as if the **-all** parameter has been supplied, which results in all the scheduled transfers, the resource monitor and scheduled transfer definitions on the agent being cleared.

 From IBM MQ 9.0.3 and IBM MQ 9.0.0 Fix Pack 2, you must, by default, specify which Managed File Transfer state to clear by passing the appropriate parameters to the **fteCleanAgent** command, as well as providing an agent name. This means that, by default, **fteCleanAgent** does not clear all in-progress and pending transfers, resource monitor definitions and scheduled transfer definitions for the agent specified. You can enable or disable this behavior by setting the `failCleanAgentWithNoArguments` property in the `command.properties` file to the appropriate value:

- By default, the value of `failCleanAgentWithNoArguments` is `true`, which means that the **fteCleanAgent** command fails to run if only the **agent_name** parameter is specified.
- If `failCleanAgentWithNoArguments` is set to `false` and only the **agent_name** parameter is specified, **fteCleanAgent** behaves in the same way as it does when you specify the **-all** parameter.

You must run the **fteCleanAgent** command on an agent that has been stopped. If you try to run the command on an agent that is currently running, you receive an error. This command does not start the agent. The **fteCleanAgent** command cleans up an agent on the system where you issue the command. You cannot clean up an agent on a remote system. To run the **fteCleanAgent** command you must have write access to the agent lock file, which is located at `MQ_DATA_PATH\mqft\logs\coordination_QMgr_name\agents\agent_name\agent.lck`

The FTEAGENT group must have GET and BROWSE authority on the following queues to run **fteCleanAgent** successfully:

- `SYSTEM.FTE.COMMAND.agent_name`
- `SYSTEM.FTE.EVENT.agent_name`
- `SYSTEM.FTE.STATE.agent_name`

See [“Restricting group authorities for MFT-specific resources”](#) on page 337 for further information about the FTEAGENT group and restricting group authorities.

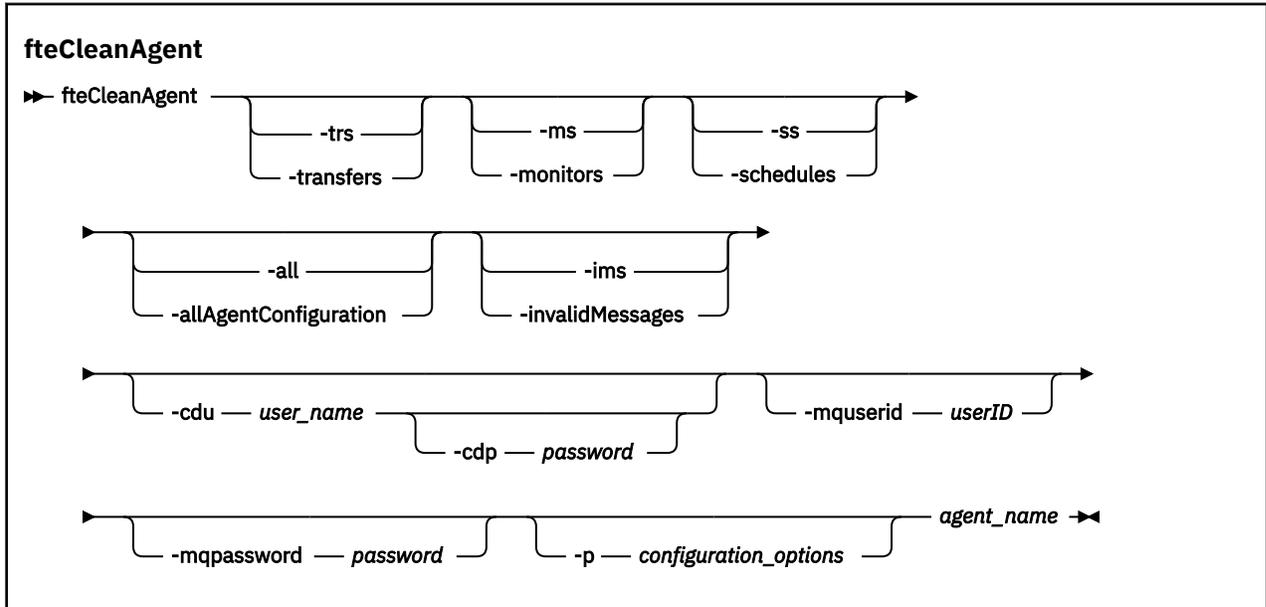
If you are running the **fteCleanAgent** command on an agent that is connected to its queue manager in bindings mode, and the agent has recently stopped running, the **fteCleanAgent** command might report

messaging problem: MQRC 2042. This MQRC occurs because a queue handle for the agent still exists in the queue manager. After a short delay the queue manager removes this handle, and you can reissue **fteCleanAgent**.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Note: When cleaning a Connect:Direct bridge agent, the user ID used to run the **fteCleanAgent** command must have read and write access to the Connect:Direct bridge agent temporary directory.

Syntax



Parameters

You can use the **fteCleanAgent** command to delete specific artifacts. For example, you can specify the **-trs** command to delete pending transfers but not change any resource monitors and scheduled transfers.

-trs or **-transfers**

Optional. Specifies that in-progress and pending transfers are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

-ms or **-monitors**

Optional. Specifies that all resource monitor definitions are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

-ss or **-schedules**

Optional. Specifies that all scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-all** or **-ims** parameters.

-all or **-allAgentConfiguration**

Optional. Specifies that all transfers, resource monitor definitions and scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-trs**, **-ss**, **-ms**, or **-ims** parameters.



Attention: You should use the **all** parameter only if no other options are available. The action of deleting transfers, resource monitor definitions, and scheduled transfer definitions can have a significant impact on your enterprise.

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

“The MFT command.properties file” on page 522

The command.properties file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

fteCreateAgent (create an MFT agent)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

You can control access to the agent. See [Restricting user authorities on MFT agent actions](#) for further information. You need to use the **-ac** parameter, and give permissions to access some queues.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Purpose

Use the **fteCreateAgent** command to create an agent. This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.agent_name
- SYSTEM.FTE.AUTHAGT1.agent_name
- SYSTEM.FTE.AUTHMON1.agent_name
- SYSTEM.FTE.AUTHOPS1.agent_name
- SYSTEM.FTE.AUTHSCH1.agent_name
- SYSTEM.FTE.AUTHTRN1.agent_name
- SYSTEM.FTE.COMMAND.agent_name
- SYSTEM.FTE.DATA.agent_name
- SYSTEM.FTE.EVENT.agent_name
- SYSTEM.FTE.REPLY.agent_name
- SYSTEM.FTE.STATE.agent_name

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc`

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues used by the agent. The MQSC commands are in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc`

Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in [The agent.properties file](#).

You might need to create a `MQMFTCredentials.xml` credentials file in order to work with your agent. A sample of this file is located in `MQ_INSTALLATION_PATH/mqft/samples/credentials/`. For more information and examples, see [“MFT credentials file format”](#) on page 785.

Important:

On UNIX platforms and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: `<MQ_DATA_PATH>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>` where `MQ_DATA_PATH` is `/var/mqm` by default.

For a re-distributable agent, this socket file is created under the directory: `<RE_DISTRIBUTABLE_DIRECTORY>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>`.

For example, if the agent name is `SRCAGENT`, the agent queue manager name is `SRCAGENTQM`, the coordination queue manager name is `COORDQM`, and the redistributable agent is running from the directory `/home/myuser/mqmft-redis`, the full path of this socket file is: `/home/myuser/mqmft-redis/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM`

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107 characters. This is particularly important with a redistributable agent where the log directory of the agent can be located in an arbitrary directory location. See the [**fteCreateEnvironment**](#) command for details on setting up the configuration directory.

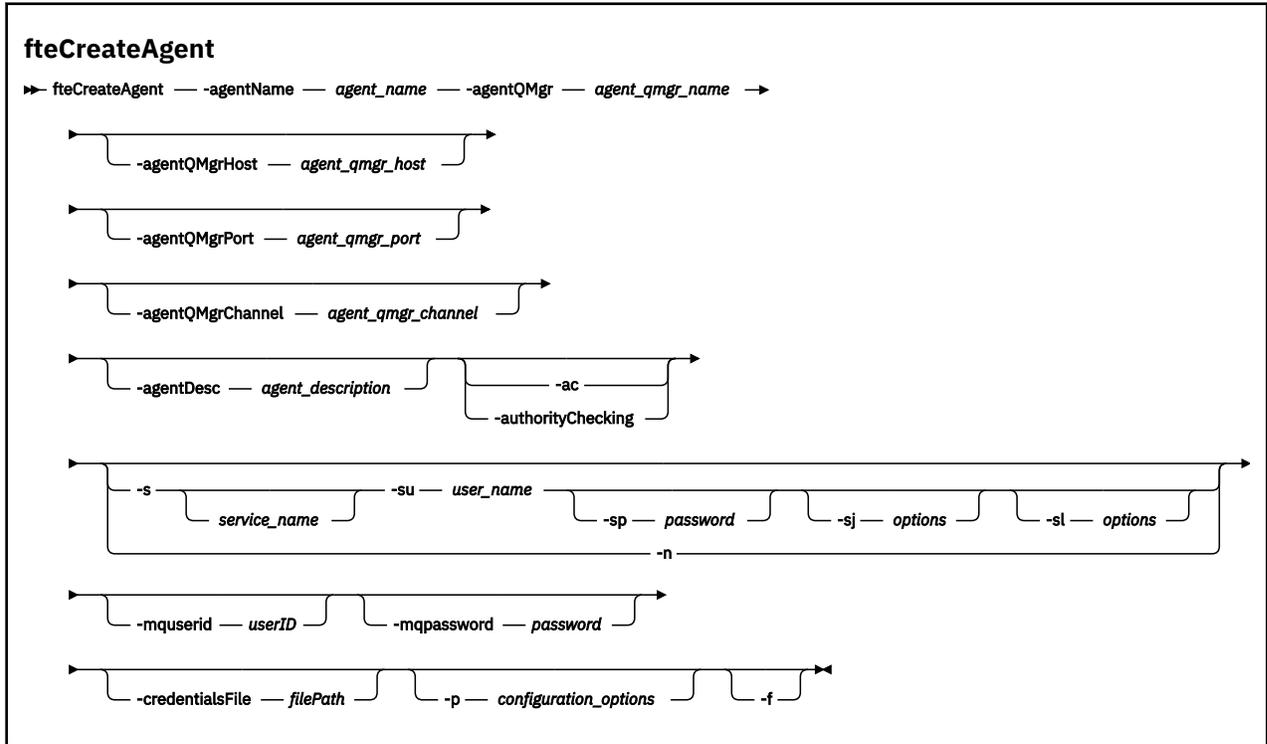
If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

```
BFGNV0159E: Failed trying to bind to socket file with FFDC
```

Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), forward-slash or back-slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double/single quotation marks or escape the special characters by using the escape sequence of the command shell.

Syntax



Parameters

-agentName (*agent_name*)

Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see [Object naming conventions](#).

-agentQMgr (*agent_qmgr_name*)

Required. The name of the agent queue manager.

-agentQMgrHost (*agent_qmgr_host*)

Optional. The host name or IP address of the agent queue manager.

-agentQMgrPort (*agent_qmgr_port*)

Optional. The port number used for client connections to the agent queue manager.

-agentQMgrChannel (*agent_qmgr_channel*)

Optional. The channel name used to connect to the agent queue manager.

-agentDesc (*agent_description*)

Optional. A description of the agent, which is displayed in IBM MQ Explorer.

-ac or -authorityChecking

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

Windows

-s (*service_name*)

Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

Windows -su (user_name)

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Guidance for running an MFT agent or logger as a Windows service](#).

Required when **-s** specified.

Windows -sp (password)

Optional (Windows only).

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

Windows -sj (options)

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (**#**) or semicolon (**;**) character. If you must embed any **#** or semicolon (**;**) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

Windows -sl (options)

Optional (Windows only). Sets the Windows service log level. Valid options are: **error**, **info**, **warn**, **debug**. The default is **info**. This option can be useful if you are having problems with the Windows service. Setting it to **debug** gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

Windows -n

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

-p (configuration_options)

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-credentialsFile (filePath)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-credentialPath (credentials_path).

This command defines the location to migrate the credential information to. This parameter can be a directory path to an existing credential file, or a directory path to a new credential file.  On z/OS platforms the credential file can be a pre-existing partitioned data set extended (PDSE). The PDSE can include existing members, or a new member for the credential file. Existing members of the PDSE must be updated to include the credential file. The format of the PDSE must be variable blocked.

-f

Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

-? or -h

Optional. Displays command syntax.

Example

In this example, AGENT3 is created with an agent queue manager QM_NEPTUNE and uses the default coordination queue manager:

```
fteCreateAgent -agentName AGENT3 -agentQMGr QM_NEPTUNE  
-agentQMGrHost myhost.ibm.com -agentQMGrPort 1415 -agentQMGrChannel CHANNEL1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

fteCreateBridgeAgent (create and configure an MFT protocol bridge agent)

The **fteCreateBridgeAgent** command creates a Managed File Transfer protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Purpose

Use the **fteCreateBridgeAgent** command to create a protocol bridge agent. For an overview of how to use the protocol bridge, see [The protocol bridge](#). This **fteCreateBridgeAgent** command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues use by the agent. The MQSC commands are in a file in the following location:

MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc.

The **fteCreateBridgeAgent** command creates a ProtocolBridgeProperties.xml XML file in the following directory:

MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name.

Users are responsible for manually creating the ProtocolBridgeCredentials.xml file, it is no longer created by the **fteCreateBridgeAgent** command.

The ProtocolBridgeCredentials.xml file allows you to define user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server and the ProtocolBridgeProperties.xml file allows you to define multiple protocol file servers so you can transfer to multiple endpoints.

There is a sample ProtocolBridgeCredentials.xml in the *MQ_INSTALLATION_PATH*/mqft/samples/credentials/ directory. For more information, see [“Protocol bridge credentials file format”](#) on page 549 and [“Protocol bridge properties file format”](#) on page 553.

If you run the **fteCreateBridgeAgent** command and specify a default protocol file server (parameter -bt), this default server is contained in the ProtocolBridgeProperties.xml file and its host name is used for the server name. With the -bt parameter, you need to specify the following parameters:

- -bh
- -btz
- -bm
- -bsl
- -bfe
- -bts

If you do not specify a default server, there are no entries in the `ProtocolBridgeProperties.xml` file; you must add at least one server manually before transfers can take place.

Managed File Transfer provides advanced agent properties that help you configure protocol bridge agents. The properties that relate to the protocol bridge start with `protocol`. These properties are described in [Advanced agent properties: Protocol bridge](#) and [Advanced agent properties: Protocol bridge agent logging](#). If you see unexpected behavior in the protocol bridge, review these `protocol` properties and ensure that you have set these properties correctly for your system.

If you see the following output from the **`fteCreateBridgeAgent`** command:

```
BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a connection attempt.
The WebSphere MQ reason code was 2058. The agent's presence will not be published.
```

it indicates that the coordination queue manager can not be contacted and provides the IBM MQ reason code for why. This information message can indicate that the coordination queue manager is currently unavailable or that you have defined the configuration incorrectly.

Important:

On UNIX platforms and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: `<MQ_DATA_PATH>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>` where `MQ_DATA_PATH` is `/var/mqm` by default.

For a re-distributable agent, this socket file is created under the directory: `<RE_DISTRIBUTABLE_DIRECTORY>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>`.

For example, if the agent name is `SRCAGENT`, the agent queue manager name is `SRCAGENTQM`, the coordination queue manager name is `COORDQM`, and the redistributable agent is running from the directory `/home/myuser/mqmft-redis`, the full path of this socket file is: `/home/myuser/mqmft-redis/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM`

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107 characters. This is particularly important with a redistributable agent where the log directory of the agent can be located in an arbitrary directory location. See the **`fteCreateEnvironment`** command for details on setting up the configuration directory.

If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

```
BFGNV0159E: Failed trying to bind to socket file with FFDC
```

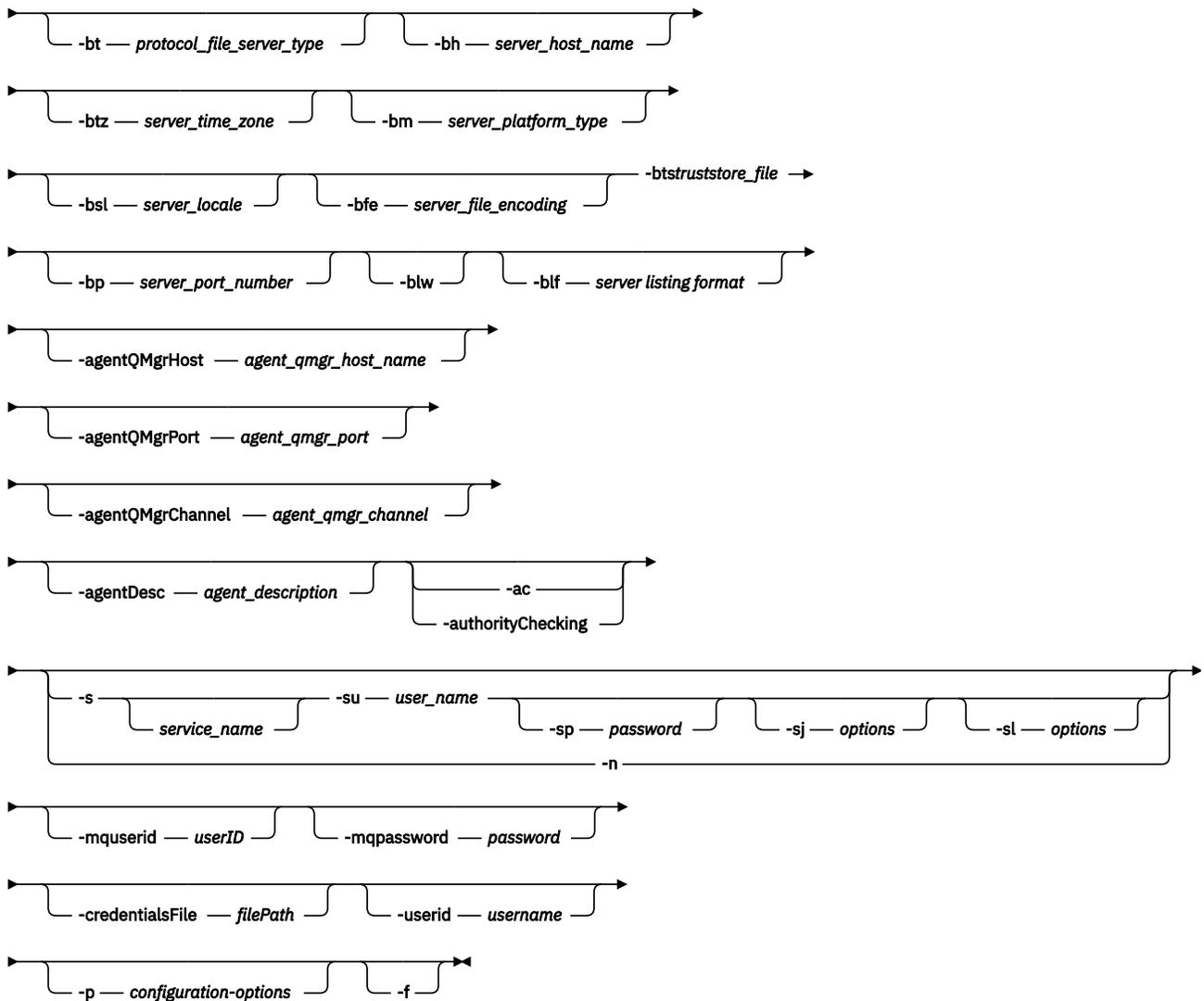
Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), forward-slash or back-slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double/single quotation marks or escape the special characters by using the escape sequence of the command shell.

Syntax

fteCreateBridgeAgent

► fteCreateBridgeAgent — -agentName *agent_name* -agentQMgr *agent_qmgr_name* →



Parameters

-agentName (*agent_name*)

Required. The name of the agent you want to create. The agent name must be unique in its administrative domain.

For more information about naming agents, see [Object naming conventions](#).

-agentQMgr (*agent_qmgr_name*)

Required. The name of the agent queue manager.

-bt (*protocol_file_server_type*)

Optional. Specifies that you want to define a default protocol file server. Specify one of the following options:

FTP

Standard FTP server

SFTP

SSH FTP server

FTPS

FTP server secured using SSL or TLS

If you do not specify this parameter, no default protocol server is defined.

-bh (server_host_name)

Required only if you also specify a default protocol file server using the **-bt** parameter. The IP host name or IP address of the protocol file server.

-btz (server_time_zone)

Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The time zone of the protocol file server. Specify the time zone in the following format: Area/Location. For example: Europe/London.

You can use the **-htz** parameter to list the possible values for **-btz**. For example:
fteCreateBridgeAgent -htz

-bm (server_platform)

Required only if you also specify a default protocol file server using the **-bt** parameter. The platform type of the protocol file server. Specify one of the following options:

UNIX

Generic UNIX and Linux platform

WINDOWS

Generic Windows platform

-bsl (server_locale)

Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The locale of the protocol file server. Specify the locale in the following format: *xx_XX*. For example: en_GB.

- *xx* is the ISO Language Code. For a list of valid values, see [Codes for the Representation of Names of Languages](#)
- *XX* is the ISO Country Code. For a list of valid values, see [Country names and code elements](#)

-bfe (server_file_encoding)

Required only if you also specify a default protocol file server using the **-bt** parameter. The character encoding format of the files stored on the protocol file server. For example: UTF-8.

You can use the **-hcs** parameter to list the possible values for **-bfe**. For example:
fteCreateBridgeAgent -hcs

-bts (truststore_file)

Required when you specify the **-bt** parameter (FTPS servers only). Specifies the path to a truststore that is used to validate the certificate presented by the FTPS server.

You can specify the **-bts** parameter only if you have also specified the FTPS option on the **-bt** parameter.

-bp (server_port)

Optional. The IP port that the protocol file server is connected to. Specify this parameter only if your protocol file server does not use the default port for that protocol. If you do not specify this parameter, Managed File Transfer uses the default port for the protocol type of file server.

-blw

Optional. Defines the protocol file server as having limited write abilities. By default, a protocol bridge agent expects the protocol file server to permit file deletion, file renaming, and file opening for append writing. Specify this parameter to indicate that the protocol file server does not permit these file actions. Instead the file server permits read from and write to file only. If you specify this parameter,

any transfers might not be recoverable if they are interrupted and might result in a failure for the file currently being transferred.

-blf (server listing format)

Optional and for FTP and FTPS servers only. Defines the server listing format of the listed file information returned from the default protocol file server. The options are as follows:

UNIX

Generic UNIX and Linux platform

WINDOWS

Generic Windows platform

To identify which format to select, use a FTP client program and perform a listing of a directory and select which format is the best fit. For example,

UNIX UNIX display the following type of listing:

```
-rwxr-xr-x 2 userid groupId 4096 2009-07-23 09:36 filename
```

Windows Windows displays the following type of listing:

```
437,909 filename
```

IBM i IBM i displays the following type of listing:

```
0S400IFS -rwxrwsrwx 3 USERID 0 8192 Mar 7 08:33 filename
```

The default is UNIX, which is the format used by most servers.

-agentQMGrHost (agent_qmgr_host)

Optional. The host name or IP address of the agent queue manager.

-agentQMGrPort (agent_qmgr_port)

Optional. The port number used for client connections to the agent queue manager.

-agentQMGrChannel (agent_qmgr_channel)

Optional. The channel name used to connect to the agent queue manager.

-agentDesc (agent_description)

Optional. A description of the agent, which is displayed in the IBM MQ Explorer.

-ac or -authorityChecking

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

Windows -s (service_name)

Optional (Windows only). Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

Windows -su (user_name)

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user

account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Guidance for running an MFT agent or logger as a Windows service](#).

Required when **-s** specified.

Windows -sp (password)

Optional (Windows only). Password for the user account set by the **-su** parameter.

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

Windows -sj (options)

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any # or semicolon (;) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. .

Windows -sl (options)

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

Windows -n

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

-p (configuration-options)

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateBridgeAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the `installation.properties` file are used. See [Configuration options](#) for more information.

-f

Optional. Forces the command to overwrite the existing configuration.

-htz

Optional. Displays a list of supported time zones that you can use as input for the **-btz** parameter.

-hcs

Optional. Displays a list of supported character sets that you can use as input for the **-bfe** parameter.

Run the **fteCreateBridgeAgent -hcs** command to list the known code pages for the JVM. This information is not available from an external source because the known code pages vary between JVMs.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-credentialsFile (filePath)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-userid (username)

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

-? or -h

Optional. Displays command syntax.

Deprecated parameters

The following parameters have been deprecated and are not supported on IBM WebSphere MQ 7.5 or on IBM WebSphere MQ File Transfer Edition 7.0.2 or later.

-brd (reconnect_delay)

Deprecated. Optional. Specifies in seconds the delay period between attempts to re-establish a lost connection with the protocol file server. The default value is 10 seconds.

-brr (reconnect_retries)

Deprecated. Optional. Specifies the maximum number of times to try again when attempting to re-establish a lost connection with the default protocol file server. When this maximum number is reached, the current file transfer is classed as failed. The default value is 2.

Examples

In this example, a new protocol bridge agent ACCOUNTS1 is created with an agent queue manager QM_ACCOUNTS and uses the default coordination queue manager. ACCOUNTS1 connects to the FTP server accountshost.ibm.com. This FTP server runs on Windows using a time zone of Europe/Berlin, a locale of de_DE, and a file encoding of UTF-8. The number of reconnect retries is 4:

```
fteCreateBridgeAgent -agentName ACCOUNTS1 -agentQMgr QM_ACCOUNTS -bt FTP
-bh accountshost.ibm.com -bm WINDOWS -btz Europe/Berlin -bsl de_DE -bfe UTF8
-agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

In this example, a new protocol bridge agent ACCOUNTS2 is created with an agent queue manager QM_ACCOUNTS and uses the default coordination manager. ACCOUNTS2 is created without a default protocol file server.

```
fteCreateBridgeAgent -agentName ACCOUNTS2 -agentQMgr QM_ACCOUNTS
```

Note: The above does not apply to Managed File Transfer Agent redistributable.

Additional customizing

If you used the **-bt** parameter (and the additional parameters that are required) there will be a default server name in the `ProtocolBridgeProperties.xml` file.

If you want to add additional ftp servers, or change the location of the credentials file, see [Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file](#).

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Use the **fteStartAgent** command to start your protocol bridge agent. For more information, see [“fteStartAgent” on page 505](#).  See also [Starting an MFT agent on z/OS](#).

Related concepts

The protocol bridge

Related reference

[“Protocol bridge credentials file format” on page 549](#)

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

[“Protocol bridge properties file format” on page 553](#)

The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

V 9.0.1

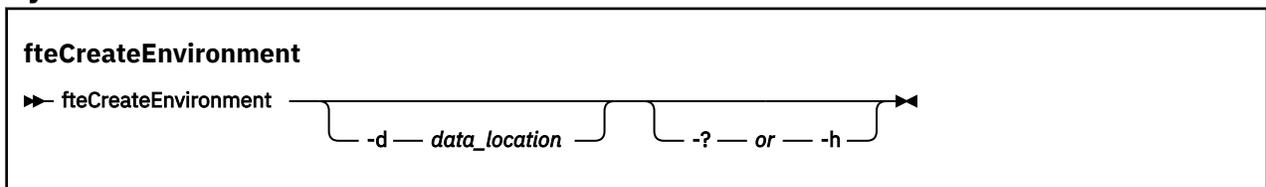
fteCreateEnvironment (set up environment for Redistributable Managed File Transfer Agent)

The **fteCreateEnvironment** command sets the environment for the configuration and transfer of files for the Redistributable Managed File Transfer Agent.

Purpose

Use the **fteCreateEnvironment** command to set up the environment for using the Redistributable Managed File Transfer Agent. You can run this command with the `-d` parameter to specify the location for the MFT Agent data files. If you do not specify the `-d` parameter, the command creates the data files in the Redistributable Managed File Transfer Agent download location and sets the data path.

Syntax



Parameters

-d (data_location)

Optional. This parameter is used for specifying the location of the data files at the time when the environment is set up.

If you do not specify this parameter, the data directory (if it does not already exist) is created in the location where the Redistributable Managed File Transfer Agent is extracted and the environment variable (`BFG_DATA`) is set for this location.

-? or -h

Optional. Displays command syntax.

Example

In this example, on Windows, the `-d` parameter specifies the location where the data folders are created:

```
fteCreateEnvironment -d C:\mftRedistributable\mftData
```

On Linux, as a prerequisite, the command has to be run on a bash shell. In a bash shell, the command can be run in various ways, and the command file needs to be sourced:

```
source Path_of_MFTZipBin/fteCreateEnvironment
```

An alternative method is:

```
. Path_of_MFTZipBin/fteCreateEnvironment
```

or, if running from the directory where the command file is present:

```
./fteCreateEnvironment
```



Attention: Note the space following the first period character (.)

Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

fteCreateCDAgent (create a Connect:Direct bridge agent)

The `fteCreateCDAgent` command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message `BFGCL0502E: You are not authorized to perform the requested operation.` and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the `mqm` group (if the `mqm` group is defined on the system).
-  Be a member of the group named in the `BFG_GROUP_NAME` environment variable (if one is named).
-  Have no value set in the `BFG_GROUP_NAME` environment variable when the command is run.

Purpose

Use the **`fteCreateCDAgent`** command to create a Connect:Direct bridge agent. This type of agent is dedicated to transferring files to and from Connect:Direct nodes. For more information, see [The Connect:Direct bridge](#). For details of the supported operating system versions for the Connect:Direct bridge, see the web page [System Requirements for IBM MQ](#).

This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- `SYSTEM.FTE.AUTHADM1.agent_name`

- SYSTEM.FTE.AUTHAGT1.agent_name
- SYSTEM.FTE.AUTHMON1.agent_name
- SYSTEM.FTE.AUTHOPS1.agent_name
- SYSTEM.FTE.AUTHSCH1.agent_name
- SYSTEM.FTE.AUTHTRN1.agent_name
- SYSTEM.FTE.COMMAND.agent_name
- SYSTEM.FTE.DATA.agent_name
- SYSTEM.FTE.EVENT.agent_name
- SYSTEM.FTE.REPLY.agent_name
- SYSTEM.FTE.STATE.agent_name

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc.`

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues belonging to the agent. The MQSC commands are in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc.`

Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in [The MFT agent.properties file](#).

The **fteCreateCDAgent** command creates two XML files in the agent properties directory. `ConnectDirectNodeProperties.xml`, which is used to define information about the remote nodes in a transfer, and `ConnectDirectProcessDefinitions.xml`, which is used to specify which user-defined Connect:Direct processes are started by transfers.

To define user names and passwords that the Connect:Direct bridge agent uses to connect to Connect:Direct nodes, you must manually create a `ConnectDirectCredentials.xml` file. Sample XML files are located in `MQ_INSTALLATION_PATH/mqft/samples/credentials/`. For more information and examples, see [“Connect:Direct credentials file format” on page 560](#).

Important:

On UNIX platforms and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: `<MQ_DATA_PATH>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>` where `MQ_DATA_PATH` is `/var/mqm` by default.

For a re-distributable agent, this socket file is created under the directory: `<RE_DISTRIBUTABLE_DIRECTORY>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>`.

For example, if the agent name is SRCAGENT, the agent queue manager name is SRCAGENTQM, the coordination queue manager name is COORDQM, and the redistributable agent is running from the directory `/home/myuser/mqmft-redis`, the full path of this socket file is: `/home/myuser/mqmft-redis/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM`

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107 characters. This is particularly important with a redistributable agent where the log directory of the agent

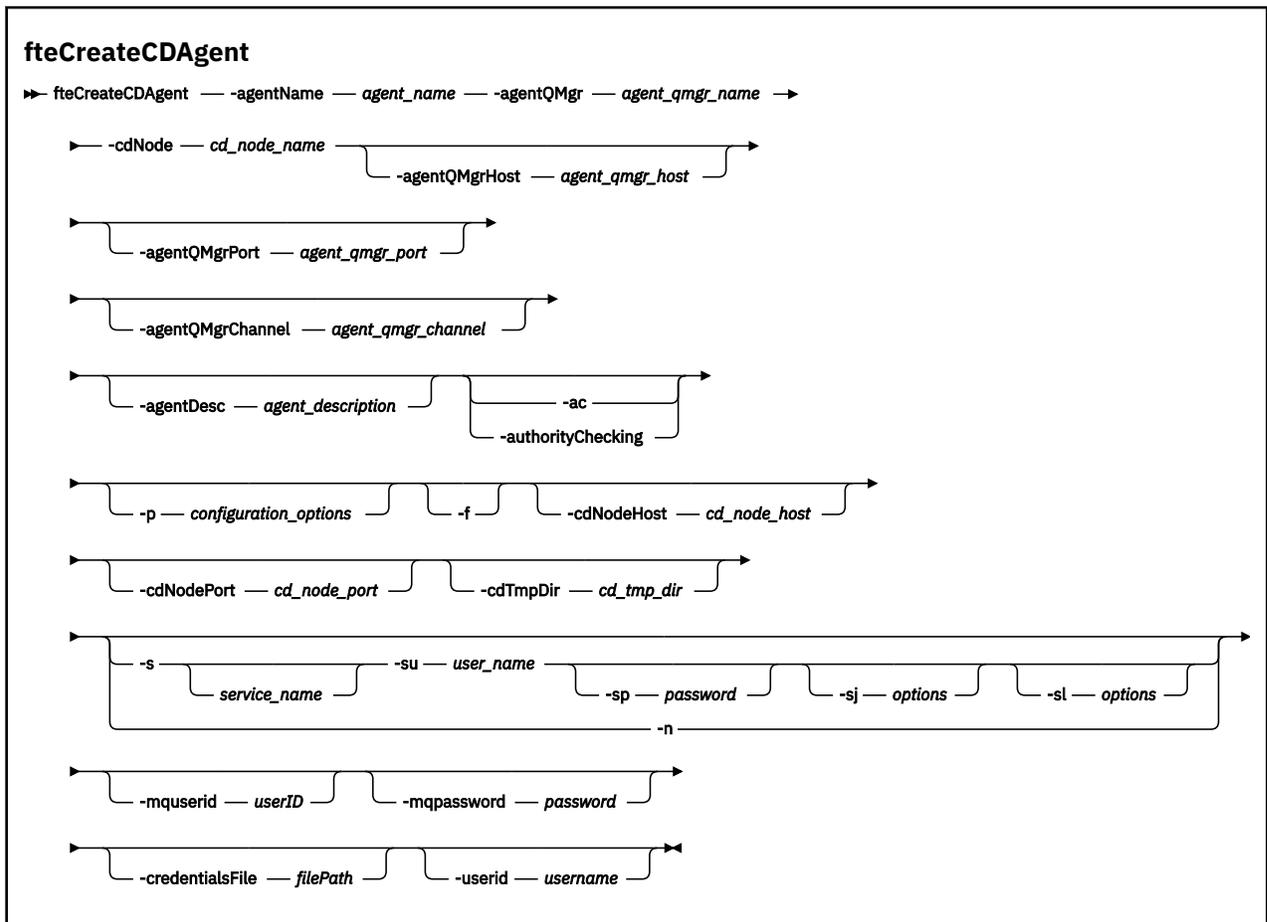
can be located in an arbitrary directory location. See the **fteCreateEnvironment** command for details on setting up the configuration directory.

If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

BFGNV0159E: Failed trying to bind to socket file with FFDC

Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), forward-slash or back-slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double/single quotation marks or escape the special characters by using the escape sequence of the command shell.



Parameters

-agentName (*agent_name*)

Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see [Object naming conventions](#).

-agentQMgr (*agent_qmgr_name*)

Required. The name of the agent queue manager.

-cdNode *cd_node_name*

Required. The name of the Connect:Direct node to use to transfer messages from this agent to destination Connect:Direct nodes. The value of this parameter is used for logging and not to specify to the Connect:Direct bridge agent which node to connect to. The values of the **-cdNodeHost** and **-cdNodePort** specify the Connect:Direct node that is part of the Connect:Direct bridge.

-agentQMgrHost (*agent_qmgr_host*)

Optional. The host name or IP address of the agent queue manager.

-agentQMgrPort (*agent_qmgr_port*)

Optional. The port number used for client connections to the agent queue manager.

-agentQMgrChannel (*agent_qmgr_channel*)

Optional. The channel name used to connect to the agent queue manager.

-agentDesc (*agent_description*)

Optional. A description of the agent, which is displayed in IBM MQ Explorer.

-ac or -authorityChecking

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

-p (*configuration_options*)

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateCDAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-f

Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

-cdNodeHost *cd_node_host_name*

Optional. The host name or IP address of the system where the Connect:Direct node, specified by the **-cdNode** parameter, is located. If you do not specify the **-cdNodeHost** parameter, a default of the host name or IP address of the local system is used.

In most cases, the Connect:Direct node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct node.

-cdNodePort *cd_node_port_name*

Optional. The port number of the Connect:Direct node that client applications use to communicate with the node that is specified by the **-cdNode** parameter. In Connect:Direct product documentation, this port is referred to as the API port. If you do not specify the **-cdNodePort** parameter, a default port number of 1363 is assumed.

-cdTmpDir *cd_tmp_directory*

Optional. The directory to be used by this agent to store files temporarily before they are transferred to the destination Connect:Direct node. This parameter specifies the full path of the directory where files are temporarily stored. For example, if **cdTmpDir** is set to /tmp then the files are temporarily placed in the /tmp directory. If you do not specify the **-cdTmpDir** parameter, the files are stored

temporarily in a directory named `cdbridge-agent_name`. This default directory is created in the location that is defined by the value of the `java.io.tmpdir` property.

Note: If you run the **fteCleanAgent** command, all files in this directory are deleted.

Windows -s (service_name)

Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

Windows -su (user_name)

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Guidance for running an MFT agent or logger as a Windows service](#).

Required when **-s** specified.

Windows -sp (password)

Optional (Windows only).

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

Windows -sj (options)

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (**#**) or semicolon (**;**) character. If you must embed any **#** or semicolon (**;**) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

Windows -sl (options)

Optional (Windows only). Sets the Windows service log level. Valid options are: `error`, `info`, `warn`, `debug`. The default is `info`. This option can be useful if you are having problems with the Windows service. Setting it to `debug` gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

Windows -n

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-credentialsFile (filePath)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-userid (username)

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

Example

In this example, a new Connect:Direct bridge agent CD_BRIDGE is created with an agent queue manager QM_NEPTUNE. The agent uses the Connect:Direct node BRIDGE_NODE to transfer files to other Connect:Direct nodes. The BRIDGE_NODE node is located on the same system as the agent and uses the default port for client connections. Files that are transferred to or from Connect:Direct are temporarily stored in the directory /tmp/cd-bridge.

```
fteCreateCDAgent -agentName CD_BRIDGE -agentQMgr QM_NEPTUNE  
                 -cdNode BRIDGE_NODE -cdTmpDir /tmp/cd-bridge
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

fteCreateLogger (create an MFT file or database logger)

Use the **fteCreateLogger** command to create a Managed File Transfer file or database logger.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Loggers on IBM i



Managed File Transfer loggers are not supported on the IBM i platform.

Purpose

The **fteCreateLogger** command provides you with the MQSC commands that you must run against your logger command queue manager to create the following logger queues:

- SYSTEM.FTE.LOG.CMD.*logger_name*
- SYSTEM.FTE.LOG.RJCT.*logger_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the logger. The MQSC commands to run are also supplied in a file in the following location:

MQ_DATA_PATH\mqft\config\coordination_qmgr\loggers*logger_name**logger_name_create.mqsc*

If you later want to delete the logger, use the **fteDeleteLogger** command.

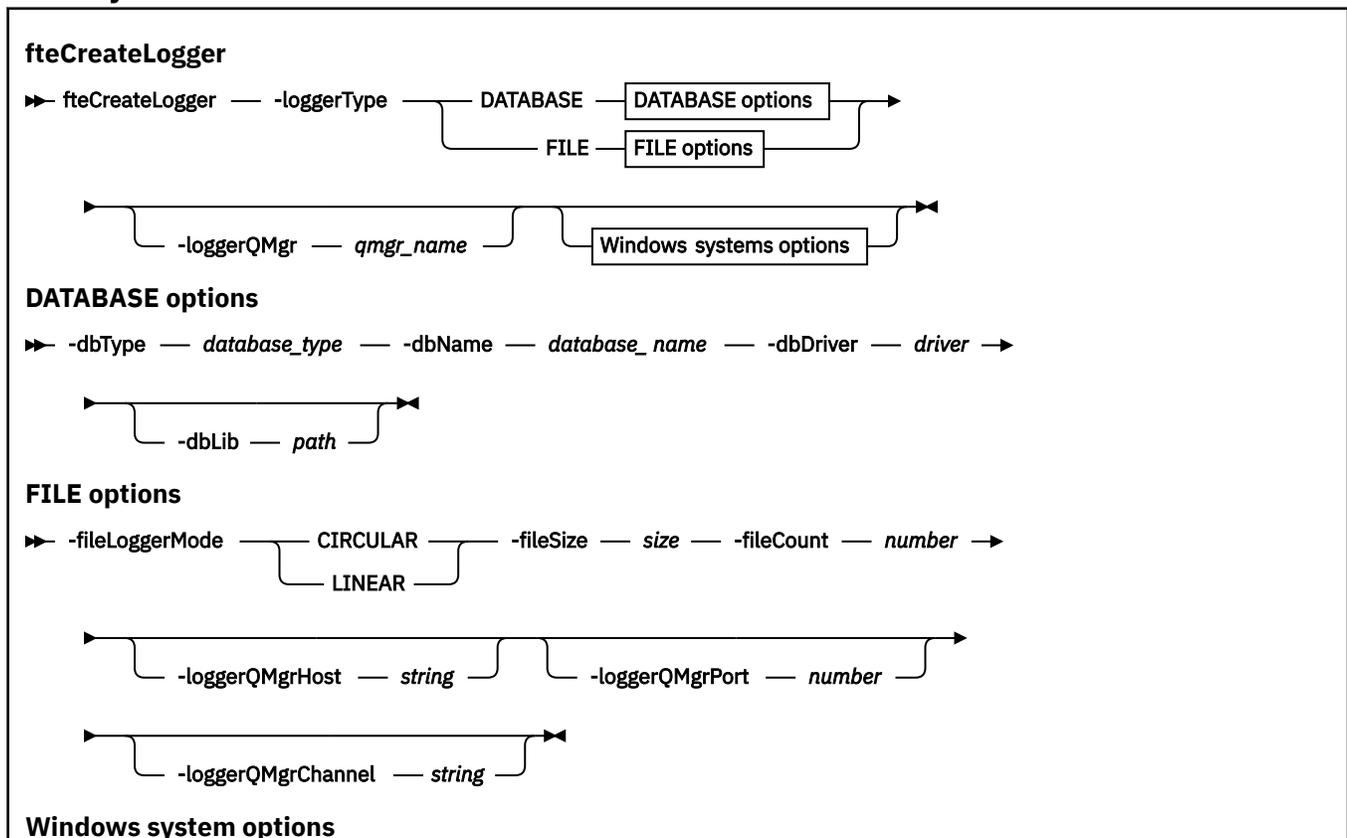
Managed File Transfer provides advanced logger properties that help you configure loggers. See, [MFT logger configuration properties](#)

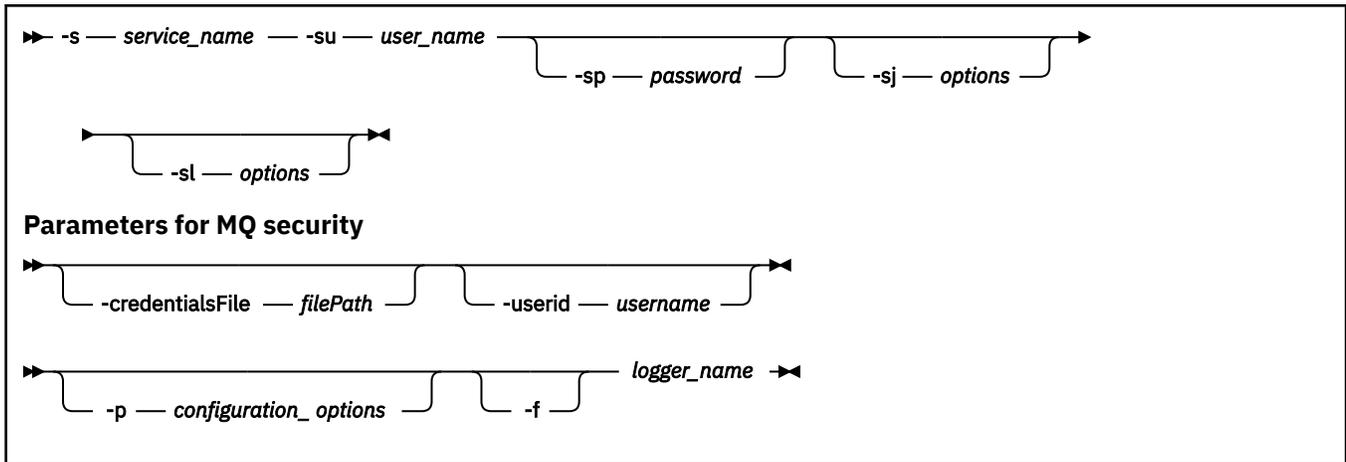
Note: If the logger you are creating is a database logger and it is not connecting to a local Db2 database, you will need to manually create a *MQMFTCredentials.xml* file. The file contains the user name and password for connecting to the database. You should use the property file, *wmqfte.database.credentials*, in the *logger.properties* file to specify the path to the *MQMFTCredentials.xml* file. A sample of this credentials file is located in *MQ_INSTALLATION_PATH/mqft/samples/credentials/*.

Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), forward-slash or back-slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double/single quotation marks or escape the special characters by using the escape sequence of the command shell.

Syntax





Parameters

-loggerType (type)

Required. Specifies where managed file transfer information will be logged. The options for type are either DATABASE, if transfer information will be logged to a database, or FILE, if the information will be logged to a file.

-loggerQMgr (qmgr_name)

Optional. Determines the queue manager to connect to in order to receive messages containing information about managed file transfers. The queue manager must be on the same system as the logger. If you do not specify the **-loggerQMgr** parameter then the coordination queue manager that is associated with the configuration options set for this logger is used as the default.

-dbType (database_type)

Required when `-loggerType` is DATABASE. Specifies the type of database management system in use for storing managed file transfer information. The options are `db2` or `oracle`

Note: You need to create tables by using SQL files. The `.sql` files are available from `MQ_INSTALLATION_PATH_/mqft/sql`:

- For Db2 databases: `ftelog_tables_db2.sql`
- For Oracle databases: `ftelog_tables_oracle.sql`

-dbName (database_name)

Required when `-loggerType` is DATABASE. The name of the database where managed file transfer information is stored. The database must be configured with the Managed File Transfer log tables.

-dbDriver (driver)

Required when `-loggerType` is DATABASE. The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file.

-dbLib (path)

Optional when `-loggerType` is DATABASE. The location of any native libraries needed by your chosen database driver.

-fileLoggerMode (mode)

Required when `-loggerType` is FILE. Specifies the type of file system in use for storing managed file transfer information. The options are `LINEAR` or `CIRCULAR`.

Option `LINEAR` means the file logger will write information to a file until that file reaches its maximum size as defined by `-filesize`. When the maximum size is reached the file logger will start a new file. Previously written files will not be deleted which allows them to be kept as a historical record of log messages. Files are not deleted when running in this mode, so the `-fileCount` will be ignored as there

is no upper limit to the number of files that can be created. As there is no upper limit when running in this mode it will be necessary to track the amount of disk space used by the log files in order to avoid running low on disk space.

Option CIRCULAR means the file logger will write information to a file until that file reaches its maximum size as defined by `-fileSize`. When the maximum size is reached the file logger will start a new file. The maximum number of files written in this mode is controlled by the value defined using the `-fileCount`. When this maximum number of files is reached the file logger will delete the first file and re-create it for use as the currently active file. If the value defined in the `-fileSize` is a fixed size byte unit, the upper limit on the disk space used in this mode will equal `fileSize x fileCount`. If the values defined in `-fileSize` are a time unit, the maximum size will depend on the throughput of log message in your system over these time periods.

For more information, see [MFT logger configuration properties](#)

-fileSize (size)

Required when `-loggerType` is FILE. The maximum size that a log file is allowed to grow to. The value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example: `-fileSize 5MB` (specifies a maximum size of 5MB), `-fileSize 2d` (specifies a maximum of 2 days worth of data).

-fileCount (number)

Required when `-loggerType` is FILE and `-fileLoggerMode` is CIRCULAR. The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of log files never exceeds the value specified in this parameter.

-loggerQMgrHost

Host name, or IP address, of the machine where the logger queue manager is running.

The default value is None.

If you do not specify the **-loggerQMgrHost** parameter, the logger is created in bindings mode.

-loggerQMgrPort

Port number where the logger queue manager is listening.

The default value is 1414.

-loggerQMgrChannel

Name of the channel used for connecting to the logger queue manager.

The default value is SYSTEM.DEF.SVRCONN.

Windows -s (service_name)

Optional (Windows systems only). Indicates that the logger is to run as a Windows service. If you do not specify `service_name`, the service is named `mqmftLogger\LOGGERQMGR`, where `LOGGER` is the logger name and `QMGR` is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Logger `LOGGER@QMGR`**.

Windows -su (user_name)

Optional (Windows only). When the logger is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the logger using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Guidance for running an MFT agent or logger as a Windows service](#).

Required when **-s** specified.

Windows -sp (password)

Optional (Windows only). Password for the user account set by the **-su** parameter.

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

Windows -sj (options)

Optional (Windows only). When the logger is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (**#**) or semicolon (**;**) character. If you must embed any (**#**) or semicolon (**;**) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

Windows -sl (options)

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

-p (configuration options)

Optional. Specifies the set of configuration options that is used to create the logger. By convention, this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f

Optional. Forces the command to overwrite the existing configuration.

(logger_name)

Required. Name of the logger to create. This is incorporated into Managed File Transfer queue names, and so must contain only letters, numbers, and the periods (**.**) and underscore characters (**_**). It is also limited to a maximum length of 28 characters.

-credentialsFile (filePath)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-userid (username)

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

-? or -h

Optional. Displays command syntax.

Examples

In this example, a circular file logger is created called filelogger1. The file logger will create a maximum of 10 files, each file being 10MB in size, using a maximum of 100MB of disk space in total:

```
fteCreateLogger -loggerType FILE -fileLoggerMode CIRCULAR -fileSize 10MB -fileCount 10
filelogger1
```

In this example, a database logger is created called `dblogger1`. The database logger connects to a Db2 database called `FTEDB`:

```
fteCreateLogger -loggerType DATABASE -dbName FTEDB -dbType DB2
                -dbDriver "C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc4.jar" dblogger1
```

In this example, a database logger is created called `dblogger1`. The database logger connects to an Oracle database called `FTEDB`:

```
fteCreateLogger -loggerType DATABASE -dbName FTEDB -dbType oracle
                -dbDriver "C:\app\oracle\product\12.1.0\dbhome_2\jdbc\lib\ojdbc7.jar" dblogger1
```

In this example, a client mode file logger is created, using the host name and default port and channel:

```
fteCreateLogger -loggerType FILE -loggerQMGr CORDQM -loggerQMGrHost cordqm.ibm.com
                -fileLoggerMode CIRCULAR -fileSize 10MB -fileCount 10 FL1
```

In this example, a client mode file logger is created, using the host name, port, and channel:

```
fteCreateLogger -loggerType FILE -loggerQMGr CORDQM -loggerQMGrHost cordqm.ibm.com
                -loggerQMGrPort 4444 -loggerQMGrChannel LOGGER_CHANNEL -fileLoggerMode CIRCULAR -fileSize 10MB
                -fileCount 10 FL1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

fteCreateMonitor: create an MFT resource monitor

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

Purpose

Use the **fteCreateMonitor** command to create and then start a new resource monitor by using a Managed File Transfer agent. For example, you can use a resource monitor in the following way: An external application puts one or more files in a known directory and when processing is complete, the external application places a trigger file in a monitored directory. The trigger file is then detected and a defined file transfer starts and copies the files from the known directory to a destination agent.

You can use the **-ox** and **-ix** parameters to export and import a resource monitor configuration to an XML file. Importing this file with the **fteCreateMonitor** command creates a new resource monitor with the same parameters as the resource monitor given in the **fteCreateMonitor** command to export to the XML file. Additionally, you can use the **-f** and **-c** parameters to overwrite a monitor configuration dynamically.

Notes:

- There is no restriction on the number of resource monitors that can be created on an agent, and all run with the same priority. Consider the implications of overlapping monitored resources, conflicting trigger conditions and how frequently the resources are polled. For more information, see [MFT resource monitoring concepts](#).
- **V9.0.4** You cannot create a resource monitor with a task definition that contains scheduled transfers. If you try to create a resource monitor with a transfer definition that points to a transfer that is scheduled to run, and repeat, at a specific time, the following message is displayed: Task

definition file contains a scheduled transfer. A scheduled transfer can not be used with a resource monitor.

- The **fteCreateMonitor** command is not supported on protocol bridge agents.

Tip: You can also use the **fteListMonitors** command to export resource monitor configurations to an XML file:

- Using the **fteListMonitors** command with the **-ox** exports the definition for a single resource monitor.
- **V 9.0.5** From IBM MQ 9.0.5, using the **fteListMonitor** command with the **-od** exports multiple resource monitor definitions to a specified directory. You can also use the **-od** option to export a single resource monitor definition to a specified directory.

For more information about the **fteListMonitors** command, see [“fteListMonitors: list MFT resource monitors”](#) on page 455.

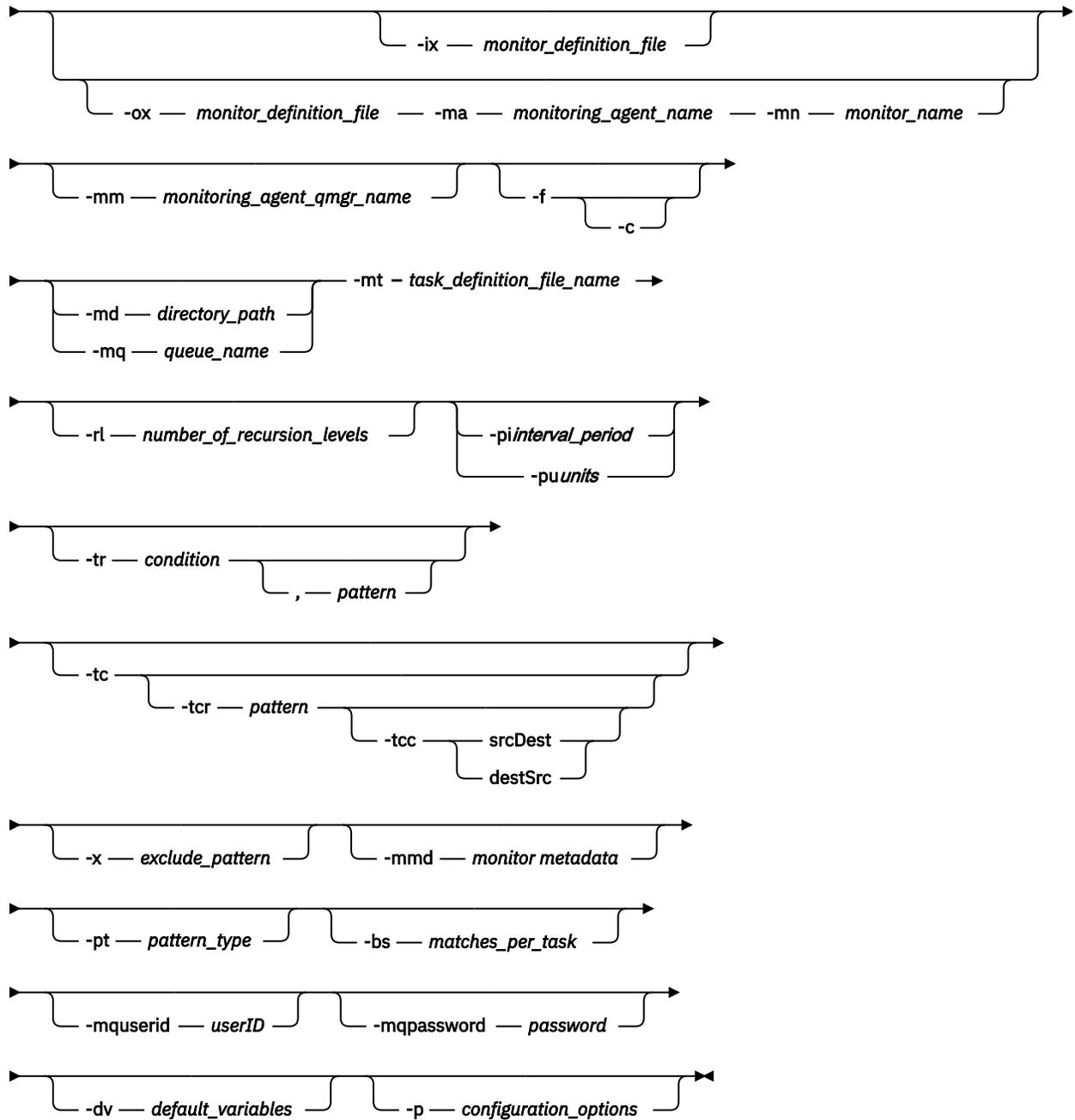
Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), forward-slash or back-slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double/single quotation marks or escape the special characters by using the escape sequence of the command shell.

Syntax

fteCreateMonitor

► fteCreateMonitor ►



Parameters

-ix (*xml_filename*)

Optional. Imports the resource monitor configuration from an XML file.

-ox (*xml_filename*)

Optional. This parameter must be specified with the **-ma** and **-mn** parameters. Exports the resource monitor configuration to an XML file.

-mn (monitor_name)

Required. The name that you assign to this monitor. The monitor name must be unique to the monitoring agent. However, you can delete a monitor and then create a monitor with the same name.

The maximum length for a resource monitor name is 256 characters. Resource monitor names are not case-sensitive. Resource monitor names that are entered in lowercase or mixed case are converted to uppercase. Resource monitor names must not contain asterisk (*), percent (%), or question mark (?) characters.

-ma (monitoring_agent_name)

Required. The name of the agent to perform the resource monitoring. This monitoring agent must be the source agent for the monitor task that you want to trigger.

-mm (monitoring_agent_qmgr_name)

The name of the queue manager that the monitoring agent is connected to. Because the monitoring agent and the source agent must be same, this queue manager is also your source agent queue manager.

Note: The **fteCreateMonitor** command connects to the command queue manager for a Managed File Transfer topology. If the command queue manager is also the agent queue manager for the monitoring agent, then this parameter is optional. Otherwise, the parameter is required.

-f

Optional. Use this parameter to overwrite a resource monitor configuration. For example, when the resource monitor name you choose already exists on the resource monitoring agent and you want to update it rather than delete and re-create a monitor with the same name. Using this parameter causes the agent to restart the monitor process.

-c

Optional. This parameter clears the history of an updated resource monitor, which causes the resource monitor to check the trigger conditions again. You can use this parameter with the **-f** parameter only.

-md (directory_path)

Optional. The absolute name of the directory path that you want to monitor. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

-mq (queue_name)

Optional. The name of the queue that you want to monitor. This queue must be on the monitoring agent queue manager. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

-mt (task_definition_file_name)

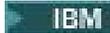
Required. The name of the XML document that contains the task definition that you want to carry out when the trigger condition is satisfied. For more information, see [“Using transfer definition files” on page 152](#). The path to the transfer definition XML document must be on the local file system that you run the **fteCreateMonitor** command from. If you do not specify a path to the file, the command looks for it in the current working directory. Unless you are using the **-ix** or **-ox** parameters, **-mt** is a required parameter.

You can use the **-gt** parameter on the **fteCreateTransfer** command to generate a template XML document that contains your file transfer request. The monitor uses the transfer template as its task definition.

V9.0.1 You can also use the transfer recovery timeout, **-rt** parameter, along with the **-gt** parameter, when you run the **fteCreateMonitor** command. You can set the amount of time in seconds during which the source agent keeps retrying to recover a transfer that is stalled. The recovery timeout parameter is then included in the XML document with the transfer definition that the monitor uses. For more information on how to set this parameter, see [fteCreateTransfer command](#).



On z/OS, you must store the task definition document in a UNIX file on z/OS UNIX System Services. You cannot store task definition documents in z/OS sequential files or PDS members.



On IBM i, you must store the task definition document in the integrated file system.

-rl (number_of_recursion_levels)

Optional. The level of monitoring recursion of the root monitoring directory that is how many levels of subdirectory to go down into. For example, in a directory structure like the following example with C:\wmqfte\monitor set as the root monitoring directory

```
C:\wmqfte\monitor
C:\wmqfte\monitor\reports
C:\wmqfte\monitor\reports\2009
C:\wmqfte\monitor\reports\2009\April
```

If you specify `-rl 2`, Managed File Transfer searches only as far down as the C:\wmqfte\monitor\reports\2009 directory and its sibling directories. The C:\wmqfte\monitor\reports\2009\April directory is ignored. By default, recursion is set to none.

-pi (interval_period)

Optional. The interval period between each monitor of a directory. The poll interval must be a positive integer value. The default value for `-pi` is 1.

-pu (units)

Optional. The time units for the monitor poll interval. If you specify the `-pu` parameter, you must also specify the `-pi` parameter. The default value for `-pu` is minutes. Specify one of the following options:

seconds

minutes

hours

days

-tr

Optional. Specifies the trigger condition that must be satisfied for the defined task to take place. If the condition is not satisfied, according to the source agent, the monitor task (for example the file transfer) is not started. A trigger condition consists of two optional parts, condition and pattern, separated by a comma. Specify one of the following formats:

-

```
condition,pattern
```

where *condition* is one of the following values:

match

For each trigger that is satisfied, the defined task is performed. `match` is the default value.

For example, if the match is `*.go` and the files `LONDON.go` and `MANCHESTER.go` are present, the task is performed for `LONDON.go` and another task is performed for `MANCHESTER.go`.

If the same trigger file is present from a previous poll (that is, the file has not been modified), this file has a not satisfied trigger condition. That is, the match trigger file must be new and must have been modified since last the poll before the defined task is performed.

noMatch

No files in the monitored directory match the pattern. That is, if *any* of the files in the monitored directory do not exist, the condition is satisfied. If no files match the trigger condition at the time

the monitor is created, the monitor starts instantly, but does not start again until a file match is found, and then removed.

noSizeChange=*n*

A minimum of one of the files in the directory matches the pattern and has a file size that does not change for *n* polling intervals. The value of *n* is a positive integer.

fileSize>=*size*

A minimum of one of the files in the directory matches the pattern and has a minimum file size greater or equal to *size*. The value *size* is a combination of an integer with an optional size unit of B, KB, MB, or GB. For example, `fileSize">"=10KB`. If you do not specify a size unit, the default size that is used is bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `fileSize` option on the command line, as shown in this example.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default value for the pattern is `*`, or match any file, and the default format is wildcard format. Use the `-pt` to specify the format of the pattern.

For example, the following trigger condition is satisfied when a file exists in the monitored directory with the suffix `.go`.

```
-tr match,*.go
```

The following trigger condition is satisfied when there are no files in the monitored directory that have the suffix `.stop`.

```
-tr noMatch,*.stop
```

You can specify *condition*,*pattern* only if you also specify the `-md` parameter.

```
condition
```

where *condition* is one of the following values:

queueNotEmpty

The monitored queue is not empty. That is, if there are *any* IBM MQ messages on the monitored queue, the condition is satisfied. A single task is run for all of the messages on the queue.

completeGroups

There is a complete group on the monitored queue. That is, if *any* of the IBM MQ message groups on the monitored queue are complete, the condition is satisfied. An individual task is run for each complete group on the queue.

If a single message that is not in a group is put on the queue, it is treated as if it is a complete group and a task is run for the single message.

You can specify *condition* only if you also specify the `-mq` parameter.

For each monitor that you create, you can specify the `-tr` parameter once only.

-tc

Optional. Indicates that the triggered file contains one or more file paths to generate a transfer request. The default format of the trigger file's contents is one file entry on each line. Specify the file paths either as *source file path* or *source file path,destination file path*. This parameter is available only for directory monitor triggers `match` and `noSizeChange`.

-tcr (pattern)

Optional. Specifies a replacement regular expression for parsing trigger files. If you specify the `-tcr` parameter, you must also specify the `-tc` parameter.

Design the pattern to parse each line entry completely with one or two capture groups. Group one defines the source file path and the optional group two defines the destination file path. This is the default behavior, which you can change using the `-tcc` parameter.

For more information and examples, see [“Using a trigger file” on page 180](#).

-tcc

Optional. Defines the regular expression capture group order.

srcDest

The default value where group one is the source file path and group two is the destination file path.

destSrc

The reverse of `srcDest`. Group one is the destination file path and group two is the source file path. Ensure that the regular expression for `destSrc` has two capture groups.

If you specify the **-tcc** parameter, you must also specify the **-tcx** parameter.

-x (exclude_pattern)

Optional. Specifies files that are excluded from the trigger pattern match. The trigger pattern is specified by the **-tr** parameter.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default format is wildcard format. Use the **-pt** parameter to specify the format of the pattern.

-mmd (monitor metadata)

Optional. Specifies the user-defined metadata that is passed to the monitor's exit points. The parameter can take one or more name pairs that are separated by commas. Each name pair consists of a *name=value*. You can use the **-mmd** parameter more than once in a command.

-pt (pattern_type)

Optional. The type of pattern that is used by the **-tr** and **-x** parameters. Valid values are:

wildcard

The patterns are evaluated as wildcard patterns. An asterisk (*) matches zero or more characters and a question mark (?) matches exactly one character. This is the default.

regex

The patterns are evaluated as Java regular expressions. For more information, see [“Regular expressions used by MFT” on page 633](#).

-bs (matches_per_task)

Optional. The maximum number of trigger matches to include in a single task. For example, if a value of 5 is specified for *matches_per_task* and nine trigger matches occur in a single poll interval, two tasks are performed. The first task corresponds to triggers 1-5 inclusive, and the second task corresponds to triggers 6-9. The default value of *matches_per_task* is 1.

The **-bs** parameter is supported only when the task definition XML that you supply to the **-mt** parameter is a `managedTransfer`. A `managedCall` is not supported with the **-bs** parameter.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password. The password is not displayed.

-dv (default_variables)

Optional. A comma-separated list of default variables that can be used in variable substitution when monitoring a queue. The values are in the format of a key-value pair. For example:

```
-dv size=medium,color=blue
```

For more information about variable substitution, see [“Customizing MFT resource monitor tasks with variable substitution” on page 172](#). You can only specify the **-dv** parameter if you have also specified the **-mq** parameter.

-? or -h

Optional. Displays command syntax.

-p (configuration_options)

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a nondefault coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this nondefault coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

Examples

In this example, a new resource monitor is created called MYMONITOR using the monitoring agent MYAGENT. Provided the trigger condition that a file larger than 5 MB is present in the directory C:\wmqfte\monitors, the file transfer that is defined in the file C:\templates\transfer_reports.xml is started. MYAGENT is also the source agent for the file transfer that is defined in C:\templates\transfer_reports.xml:

```
fteCreateMonitor -ma MYAGENT -md C:\wmqfte\monitors -mn MYMONITOR -mt C:\templates\transfer_reports.xml -tr fileSize">"=5MB,*go
```

In this example, a resource monitor called MONITOR1 using the agent AGENT1 is created to transfer files greater than 5 MB and is exported to the XML file monitor.xml.

```
fteCreateMonitor -ox monitor.xml -ma AGENT1 -mn MONITOR1 -mt task.xml -tr "fileSize>=5MB,*zip"
```

Then the XML file is imported and changed to exclude any files greater than 10MB.

```
fteCreateMonitor -ix monitor.xml -x "fileSize>=10MB,*zip" -f
```

In this example, a new resource monitor is created called MYMONITOR using the agent MYAGENT.

```
fteCreateMonitor -ma MYAGENT -md c:\wmqfte -mn MYMONITOR -mt c:\templates\transfer_reports.xml -tr "fileSize>=5MB,*go"
```

However the trigger is initially incorrectly set to monitor c:\wmqfte rather than c:\wmqfte\monitors. The **fteCreateMonitor** request is immediately reissued with the monitor directory corrected and the **-f** (overwrite) and **-c** (clear history) parameters used to update the monitor.

```
fteCreateMonitor -ma MYAGENT -md c:\wmqfte\monitors -mn MYMONITOR -mt c:\templates\transfer_reports.xml -tr "fileSize>=5MB,*go" -f -c
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.

fteCreateTemplate: create new file transfer template

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

Purpose

Use the **fteCreateTemplate** command to create a file transfer template that stores your transfer details until you want to use them at a later date. Use transfer templates to store common file transfer settings for repeated or complex transfers. After you have created a transfer template, submit the template using the IBM MQ Explorer. You cannot submit a transfer template from the command line.

The transfer template that you create using the **fteCreateTemplate** command is not the same as the XML message that you create using the **-gt** parameter on the **fteCreateTransfer** command. You cannot use the two different types of template interchangeably.

You can run the **fteCreateTemplate** command from any system that can connect to the IBM MQ network and then route to the coordination queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured the Managed File Transfer component on this system to communicate with the IBM MQ network.

This command uses the `command.properties` file to connect to the command queue manager for the Managed File Transfer topology. If the `command.properties` file contains the **connectionQMgrHost** property, then the command connects to the command queue manager using the CLIENT transport. Otherwise, the command connects to the command queue manager using the BINDINGS transport. If the `command.properties` file does not exist, the command will fail and generate the following error:

```
BFGCL0491E: Missing or corrupt command.properties file. Use the fteSetupCommands
command to correct this condition. Additional information might be contained in this
exception BFGUB0009E: The following required property file is missing:
"MQ_DATA_PATH\mqft\coordination\coordination_qmgr_name\command.properties"
```

For more information, see [The command.properties file.](#)

You can specify multiple source files for a file transfer but only one destination agent; transferring one file to multiple destination agents is not supported. However, you can transfer multiple source files to multiple destination files on a single destination agent.

For guidance about how to transfer files, see [“Guidelines for transferring files”](#) on page 604.

Special characters

Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect.  For example, fully qualified data set names that contain single quotation marks and source specifications that contain asterisk characters might be interpreted by the command shell rather than being passed through in the transfer request. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks as shown in the final two examples [“Examples”](#) on page 416, or escape the special characters using the escape sequence of the command shell.

Relative paths

The **fteCreateTemplate** command supports the use of relative file paths. On distributed systems  and z/OS UNIX System Services by default paths are considered to be relative to the home directory of the user that the agent is running as. To change the directory that path names are evaluated relative to, set the `transferRoot` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRoot=directory_name
```

You must escape Windows paths or write them in UNIX format. For example, specify `C:\TransferRoot` as `C:\\TransferRoot` or `C:/TransferRoot`.

 On z/OS, by default the user name that the agent is currently running under is added as a high-level qualifier prefix to data set specifications that have not been fully qualified. For example: `//`

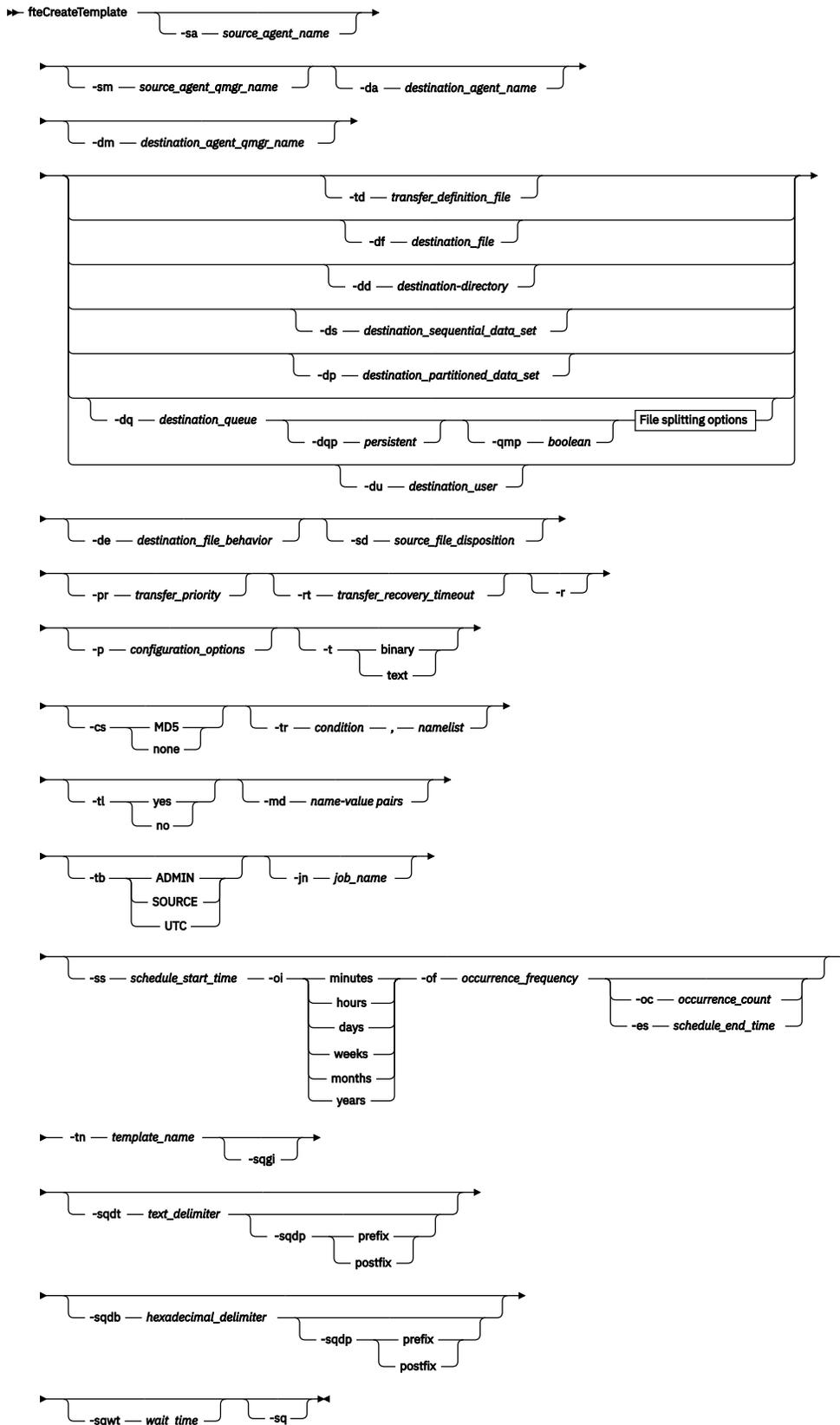
ABC.DEF. To change the value that is added as a prefix to the data set name, set the `transferRootHLQ` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRootHLQ=prepend_value
```

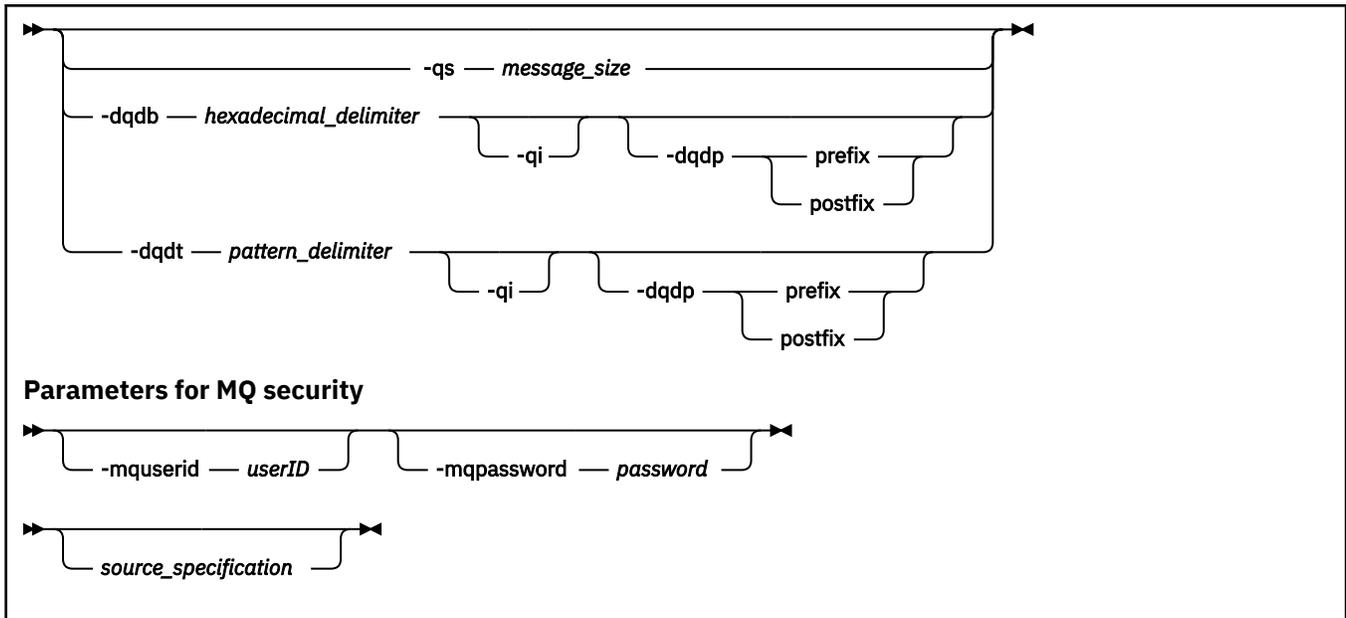
 However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name.

Syntax

fteCreateTemplate



File splitting options



Parameters

-sa *source_agent_name*

Optional. The name of the agent that the source file is transferred from. If you do not specify this agent name when you create the template, you must specify the source agent name when you use the template.

-sm *source_agent_qmgr_name*

Optional. The name of the queue manager that the source agent is connected to.

If you do not specify the **-sm** parameter, the queue manager used is determined by the set of configuration options in use, based on the source agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the agent `.properties` file for the source agent cannot be found.

-da *destination_agent_name*

Optional. The name of the agent that the file is transferred to. If you do not specify the destination agent name when you create the template, you must specify the destination agent name when you use the template.

-dm *destination_agent_qmgr_name*

Optional. The name of the queue manager that the destination agent is connected to.

If you do not specify the **-dm** parameter, the queue manager used is determined by the set of configuration options in use, based on the destination agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the agent `.properties` file for the destination agent cannot be found.

-td *transfer_definition_file*

Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-td** parameter, you cannot specify source files, or specify the **-df**, **-dd**, **-ds**, **-dp**, **-dq**, **-du**, **-sd**, **-r**, **-de**, **-t**, or **-cs** parameters.

The **ftCreateTemplate** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

z/OS On z/OS, you must store the transfer definition file in a UNIX file on z/OS UNIX System Services. You cannot store transfer definition files in z/OS sequential files or PDS members.

IBM i On IBM i, you must store the transfer definition file in the integrated file system.

For more information, see [Using transfer definition files](#).

-df destination_file

Optional. The name of the destination file. Specify a file name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format *connect_direct_node_name:file_path*. The Connect:Direct bridge agent accepts only file paths that are specified in this format. **z/OS** If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-df** parameter, you cannot specify the **-td**, **-dd**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

-dd destination_directory

Optional. The name of the directory the file is transferred to. Specify a directory name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format *connect_direct_node_name:directory_path*. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-dd** parameter, you cannot specify the **-td**, **-df**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

z/OS -ds destination_sequential_data_set

z/OS only. Optional. The name of the sequential data set or PDS member that files are transferred into. Specify a sequential data set name or a partitioned data set member.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-ds** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dq**, **-du**, or **-dp** parameters because these parameters are mutually exclusive.

The syntax for the data set name is as follows:

```
//data_set_name{;attribute;...;attribute}
```

or

```
//pds_data_set_name(member_name){;attribute;...;attribute}
```

That is, a data set name specifier prefixed with // and optionally followed by a number of attributes separated by semicolons.

If the data set is located at a Connect:Direct node, you must prefix the data set name with the node name. For example:

```
CD_NODE1:// 'OBJECT.LIB' ;RECFM(F,B);BLKSIZE(800);LRECL(80)
```

If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite. For more information about data set

transfers to or from Connect:Direct nodes, see [“Transferring data sets to and from Connect:Direct nodes”](#) on page 609.

For transfers that only involve Managed File Transfer agents, if the data set name part is enclosed by single quotation mark characters, it specifies a fully qualified data set name. If the data set name is not enclosed by single quotation mark characters, the system adds the default high-level qualifier for the destination agent (either the value for the `transferRootHLQ` agent property or the user ID that the agent runs under, if you have not set `transferRootHLQ`).

Note:  However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name. This is the case even if the data set name is enclosed by single quotation mark characters.

The data set attributes are used either to create a data set or to ensure that an existing data set is compatible. The specification of data set attributes is in a form suitable for BPXWDYN (see [Requesting dynamic allocation](#) for more information). When the agent is to create a destination data set, the following BPXWDYN attributes are automatically specified: `DSN(data_set_name) NEW CATALOG MSG(numeric_file_descriptor)`, where *numeric_file_descriptor* is a file descriptor generated by Managed File Transfer. For a data set to data set transfer, the attributes of RECFM, LRECL, and BLKSIZE from the source are selected for a new destination data set. Note the SPACE setting for a new destination data set is not set by Managed File Transfer and system defaults are used. Therefore, you are recommended to specify the SPACE attribute when a new data set is to be created. You can use the `bpxwdynAllocAdditionalProperties` property in the `agent.properties` file to set BPXWDYN options that apply to all transfers. For more information, see [“The MFT agent.properties file”](#) on page 526.

Some BPXWDYN options must not be specified when using the `fteCreateTemplate` command, the `fteCreateTransfer` command or the `bpxwdynAllocAdditionalOptions` property in the `agent.properties` file. For a list of these properties, see [“BPXWDYN properties you must not use with MFT”](#) on page 616.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as the source data set or to the default values when the source is a file. The attributes of an existing tape data set are ignored.

The `-ds` parameter is not supported when the destination agent is a protocol bridge agent.

-dp destination_partitioned_data_set

z/OS only. Optional. The name of the destination PDS that files are transferred into. Specify a partitioned data set name. If a PDS is created as a result of the transfer, this PDS is created as a PDSE by default. You can override the default by specifying `DSNTYPE=PDS`.

One of the `-td`, `-df`, `-dd`, `-ds`, `-dq`, `-du`, and `-dp` parameters is required. If you specify the `-dp` parameter, you cannot specify the `-td`, `-dd`, `-df`, `-dq`, `-du`, or `-ds` parameters because these parameters are mutually exclusive.

The syntax for the PDS data set name is as follows:

```
//pds_data_set_name{;attribute;...;attribute}
```

The syntax for the data set name is the same as described for the `-ds` (`destination_sequential_data_set`) parameter. All the syntax details for specifying data sets that are located on Connect:Direct nodes also apply to the `-dp` parameter. If the destination agent is a Connect:Direct bridge agent, you must also specify the `-de` parameter with a value of `overwrite`.

The `-dp` parameter is not supported when the destination agent is a protocol bridge agent.

-du destination_user

Optional. The name of the user whose destination file space the files are transferred into. .

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-du** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-dq**, or **-ds** parameters because these parameters are mutually exclusive.

The **-du** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent.

-dq destination_queue

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name, the destination agent queue manager name is used if you have not set the enableClusterQueueInputOutput agent property to true. If you have set the enableClusterQueueInputOutput agent property to true, the destination agent uses standard IBM MQ resolution procedures to determine where the queue is located. You must specify a valid queue name that exists on the queue manager.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-dq** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

-dqp persistent

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

true

Writes persistent messages to the destination queue. This is the default value.

false

Writes non-persistent messages to the destination queue.

qdef

The persistence value is taken from the DefPersistence attribute of the destination queue.

You can only specify the **-dqp** parameter if you have also specified the **-dq** parameter.

-qmp boolean

Optional. Specifies whether the first message written to the destination queue by the transfer has IBM MQ message properties set. The valid options are as follows:

true

Sets message properties on the first message created by the transfer.

false

Does not set message properties on the first message created by the transfer. This is the default value.

You can only specify the **-qmp** parameter if you have also specified the **-dq** parameter. For more information, see [“MQ message properties set by MFT on messages written to destination queues” on page 651](#)

-qs message_size

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST_MSG_IN_GROUP flag set. The size of the messages is specified by the value of *message_size*. The format of *message_size* is *lengthunits*, where *length* is a positive integer value and *units* is one of the following values:

B

Bytes. The minimum value allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

K

This is equivalent to 1024 bytes.

M

This is equivalent to 1048576 bytes.

If you specify the value `text` for the **-t** parameter and the file is in a double byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can only specify the **-qs** parameter if you have also specified the **-dq** parameter. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdb hexadecimal_delimiter

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a hexadecimal byte as a delimiter is `xNN`, where `N` is a character in the range `0-9` or `a-f`. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: `x3e,x20,x20,xbf`.

You can only specify the **-dqdb** parameter if you have also specified the **-dq** parameter and the transfer is in binary mode. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdt pattern

Optional. Specifies the regular expression to use when splitting a text file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a regular expression as a delimiter is a regular expression enclosed in parentheses, (*regular_expression*). The value of this parameter is evaluated as a Java regular expression. For more information, see [“Regular expressions used by MFT” on page 633](#).

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior using the `maxDelimiterMatchLength` agent property. For more information, see [“Advanced agent properties” on page 527](#).

You can only specify the **-dqdt** parameter if you have also specified the **-dq** parameter and the value `text` for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdp

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can only specify the **-dqdp** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

prefix

The delimiters are expected at the beginning of each line.

postfix

The delimiters are expected at the end of each line. This is the default option.

-qi

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. If **-qi** is specified, the delimiter is included at the end of the message that contains the file data preceding the delimiter. By default the delimiter is not included in the messages.

You can only specify the **-qi** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

-de destination_file_behavior

Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

error

Reports an error and the file is not transferred. This is the default value.

overwrite

Overwrites the existing destination file.

If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive.

-sd source_file_disposition

Optional. Specifies the action that is taken on a source file when that source file has successfully been transferred to its destination. The valid options are as follows:

leave

The source files are left unchanged. This is the default value.

delete

The source file is deleted from the source system after the source file is successfully transferred.

 On z/OS, if the source is a tape data set and you specify the delete option, the tape is remounted to delete the data set. This behavior is because of the behavior of the system environment.

If the source is a queue and you specify the leave option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the delete option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

- If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the delete option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [“Submitting a user-defined Connect:Direct process from a file transfer request” on page 229](#).
- If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

-pr transfer_priority

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

This value matches the message priority value used by IBM MQ, see [Getting messages from a queue: priority](#) for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your IBM MQ message traffic to take priority.

-rt transfer_recovery_timeout

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

-1

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

0

The agent stops the file transfer as soon as it enters recovery.

>0

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
-rt 21600
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this parameter is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a property to the [The agent.properties file](#).

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to create the transfer template. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-r

Optional. Recursively transfer files in subdirectories when *source_specification* contains wildcard characters. When Managed File Transfer is presented with a wildcard character as a *source_specification*, any directories that match the wildcard character are transferred only if you have specified the **-x** parameter. When *source_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how Managed File Transfer handles wildcard characters, see [Using wildcard characters](#)

If you specify the **-x** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

-t

Optional. Specifies the type of file transfer: binary mode or text mode.

binary

The data in the file is transferred without any conversion. This is the default value.

text

The code page and end-of-line characters of the file are converted. The exact conversions performed depend on the operating systems of the source agent and destination agent.

 For example, a file transferred from Windows to z/OS has its code page converted from ASCII to EBCDIC. When a file is converted from ASCII to EBCDIC, the end-of-line characters are converted from ASCII carriage return (CR) and line feed (LF) character pairs to an EBCDIC new line (NL) character.

 For more information about how z/OS data sets are transferred, see [Transferring files and data sets between z/OS and distributed systems](#) and [Transferring between data sets](#).

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

-cs

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

MD5

Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, Managed File Transfer computes MD5 checksums for all file transfers.

none

No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to none and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the none option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

-tr

Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

```
condition,namelist
```

where *condition* is one of the following values:

file=exist

A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

file!=exist

A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

filesize>=size

A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. The value of *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">"=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the source system. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks.

You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

Note: To continually monitor a resource for a trigger condition to be true, you are recommended to use [resource monitoring](#). You can create a resource monitor using the [fteCreateMonitor](#) command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTemplate -tn JUPITER_AGENT_TRIGGER_TEST_TEMPLATE -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm
QM_NEPTUNE
-tr file=exist,C:\export\A.txt,C:\export\B.txt
-tr filesize">"=1GB,C:\export\A.txt,C:\export\B.txt
-df C:\import\file1.doc C:\export\file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

-tl

Optional. Specifies whether trigger failures are logged. Specify one of the following options:

yes

Log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

no

No log entries are created for failed triggered transfers.

-md

Optional. Specifies the user-defined metadata that is passed to the exit points of the agent. The **-md** parameter can take one or more name-value pairs separated by commas. Each name pair consists of *name=value*. You can use the **-md** parameter more than once in a command.

 On z/OS, spaces represent delimiters so you must use underscores to separate values. For example, use `kw=text1_text2_text3` rather than `kw="text1 text2 text3"`

-tb

Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

admin

The start and end times used for the scheduled transfer are based on the time and date of the system used by the administrator. This is the default value.

source

The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

UTC

The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

-jn *job_name*

Optional. A user-defined job name identifier that is added to the log message when the transfer has started.

-ss *schedule_start_time*

Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time using the 24-hour clock:

```
yyyy-MM-ddThh:mm  
hh:mm
```

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

-oi

Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

minutes

hours

days
weeks
months
years

-of occurrence_frequency

Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

-oc occurrence_count

Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count has been met, the scheduled transfer is deleted.

Specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

-es schedule_end_time

Optional. The time and date that a repeating scheduled transfer ends.

You must specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

Use one of the following formats to specify the end time and date. Specify the time using the 24-hour clock:

```
yyyy-MM-ddThh:mm  
hh:mm
```

-tn template_name

Required. The name of the template that you want to create. Use a descriptive string that allows you to select the correct template for transfers at a later date. There is no specific limit to the length of this string, but be aware that excessively long names might not be displayed properly in some user interfaces.

Do not create multiple templates with the same name.

-sqgi

Optional. Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

You can only specify the **-sqgi** parameter if you have also specified the **-sq** parameter.

-sqdt text_delimiter

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `-sqdt \u007d\n`.

You can only specify the **-sqdt** parameter if you have also specified the **-sq** parameter and the value `text` for the **-t** parameter.

-sqdb hexadecimal_delimiter

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by x. Multiple bytes must be comma-separated. For example, -sqdb x08,xA4.

You can only specify the **-sqdb** parameter if you have also specified the **-sq** parameter. You cannot specify the **-sqdb** parameter if you have also specified the value `text` for the **-t** parameter.

-sqdp

Optional. Specifies the position of insertion of source text and binary delimiters. You can only specify the **-sqdp** parameter if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

prefix

The delimiters are inserted at the start of each message

postfix

The delimiters are inserted at the end of each message. This is the default option.

-sqwt wait_time

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to be put on the queue
- If the **-sqgi** parameter was specified, for a complete group to be put on the queue

If neither of these conditions are met within the time specified by *wait_time*, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.

You can only specify the **-sqwt** parameter if you have also specified the **-sq** parameter.

-sq

Optional. Specifies that the source of a transfer is a queue.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

source_specification

Required if you have specified one of the **-df**, **-dd**, **-dp**, **-dp**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source_specification*.

- If you have not specified the **-sq** parameter, *source_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications are space delimited. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters in WMQFTE, see [“Using wildcard characters with MFT” on page 627](#). You can escape asterisks that are part of the file specification by using two asterisk characters (**) in the file specification.

To transfer files containing spaces in their file names, place double quotation marks around the file names that contain spaces. For example to transfer file `a b.txt` to file `c d.txt` specify the following text as part of the **fteCreateTemplate** command:

```
-df "c d.txt" "a b.txt"
```

Each file specification must be in one of the following formats:

File names

The name of a file, expressed using the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.

Directories

The name of a directory, expressed using the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.

For example, to copy the contents of DIR1 to DIR2 only, specify DIR1/* DIR2

Sequential data set

(z/OS only). The name of a sequential data set or partitioned data set member. Denote data sets by preceding the data set name with two forward slash characters (//).

Partitioned data set

(z/OS only). The name of a partitioned data set. Denote data set names by preceding the data set name with two forward slash characters (//).

File name or directory at a Connect:Direct node

(Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (:), and a file or directory path on the system that is hosting the Connect:Direct node. For example, `connect_direct_node_name:file_path`.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

Note: Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

- If you have specified the **-sq** parameter, *source_specification* is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

```
QUEUE_NAME
```

The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

-? or -h

Optional. Displays command syntax.

Examples

In this example, a transfer template called `payroll accounts monthly report template` is created. When submitted, this template transfers any file with the extension `.xls` from the agent `PAYROLL1` to the agent `ACCOUNTS` in the directories specified:

```
fteCreateTemplate -tn "payroll accounts monthly report template" -sa PAYROLL -sm QM_PAYROLL1 -da  
ACCOUNTS  
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

In this example, a transfer template called `jupiter_neptune_sched_template` is created. When submitted, the template transfers the file `originalfile.txt` from the system where `QM_JUPITER` is located to the system where `QM_NEPTUNE` is located. The file transfer is scheduled to take place at

09:00 based on the system time of the system where the source agent is located and occurs every two hours four times:

```
fteCreateTemplate -tn jupiter_neptune_sched_template -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tb source -ss 09:00 -oi hours -of 2 -oc 4
-df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, a transfer template called `jupiter neptune trigger template` is created. When the template is submitted, the file `originalfile.txt` is transferred from AGENT1 to AGENT2, on condition that the file `A.txt` exists on AGENT1:

```
fteCreateTemplate -tn "jupiter neptune trigger template" -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm
QM_NEPTUNE
-tr file=exist,C:\export\A.txt -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

z/OS In this example, a template called `ascii_ebcdic_template` is created. When the template is submitted, the file `originalfile.txt` is transferred from the system where AGENT1 is located to a data set `/'USERID.TRANS.FILE.TXT'` on the system where AGENT2 is located. Text mode has been selected to convert data from ASCII to EBCDIC.

```
fteCreateTemplate -tn ascii_ebcdic_template -t text -sa AGENT1 -da AGENT2
-ds "///TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);
SPACE(5,1)" C:\export\originalfile.txt
```

z/OS In this example, a template called `ebcdic_ascii_template` is created. When the template is submitted, a member of a fully qualified data set on the system where AGENT1 is located is transferred to a file on the system where AGENT2 is located. Text mode has been selected to convert the file from EBCDIC to ASCII.

```
fteCreateTemplate -tn ebcdic_ascii_template -t text -sa AGENT1 -da AGENT2 -df /tmp/IEEUJV.txt
"///'SYS1.SAMPLIB(IEEUJV)'"
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.

fteCreateTransfer: start a new file transfer

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Purpose

Use the **fteCreateTransfer** command to create and then start a new file transfer from a Managed File Transfer agent.

Note: File transfers can only take place between agents within the same Managed File Transfer topology.

For guidance about how to transfer files, see [“Guidelines for transferring files”](#) on [page 604](#). For the z/OS platform, you can transfer text files, data sets, and generation data groups (GDGs).

You can run the **fteCreateTransfer** command from any system that can connect to the IBM MQ network and then route to the source agent queue manager. Specifically, for the command to run, you

must install a Managed File Transfer component (either Service or Agent) on this system and configure the Managed File Transfer component on this system to communicate with the IBM MQ network.

This command uses a properties file called `command.properties` to connect to the IBM MQ network. If the `command.properties` file does not contain property information, a bindings mode connection is made to the default queue manager on the local system. If the `command.properties` file does not exist, an error is generated. For more information, see [“The MFT command.properties file” on page 522](#).

You can specify multiple source files for a file transfer but they must originate from a single source agent and terminate at a single destination agent. Transferring a single source file to multiple destination files on the same agent or multiple different agents is not supported within a single transfer. Ant scripting can be used to send the same source file to multiple destinations at one or more agents. For more information, see [“Using Apache Ant with MFT” on page 241](#).

Special characters

Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified data set names that contain single quotation marks, and source specifications that contain asterisk characters, might be interpreted by the command shell rather than being passed through in the transfer request. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks or escape the special characters by using the escape sequence of the command shell.

Relative paths

The **fteCreateTransfer** command supports the use of relative file paths. For the following platforms, by default, paths are considered to be relative to the home directory of the user that the agent is running as:

- **Multi** Multiplatforms
- **z/OS** UNIX System Services on z/OS

To change the directory that path names are evaluated relative to, set the `transferRoot` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRoot=directory_name
```

Windows For example, specify `C:\TransferRoot` as `C:\\TransferRoot` or `C:/TransferRoot`.

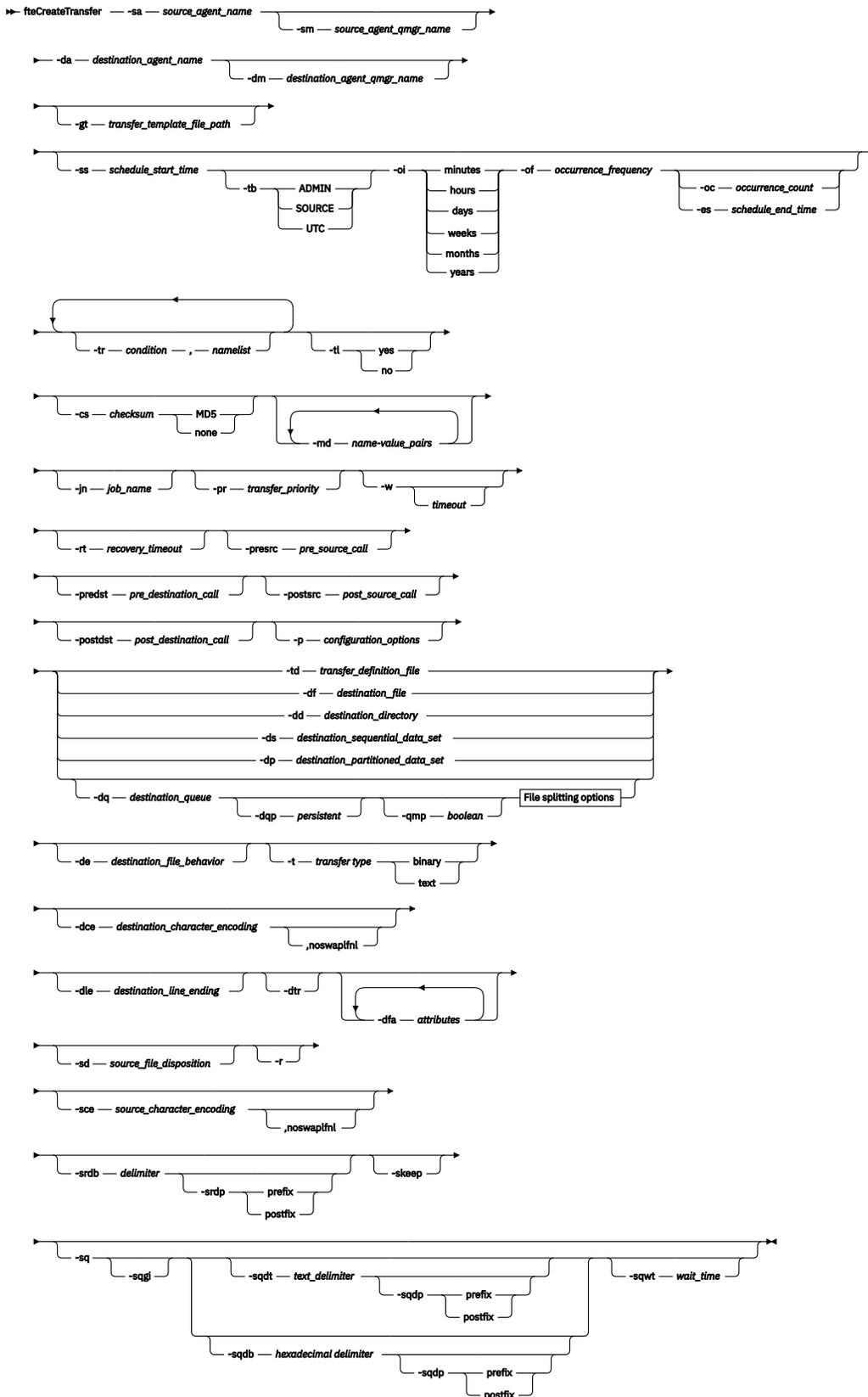
z/OS On z/OS, by default the user name that the agent is running under is added as a high-level qualifier prefix to data set specifications that have not been fully qualified. For example: `//ABC.DEF`. To change the value that is added as a prefix to the data set name, set the `transferRootHLQ` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRootHLQ=prepend_value
```

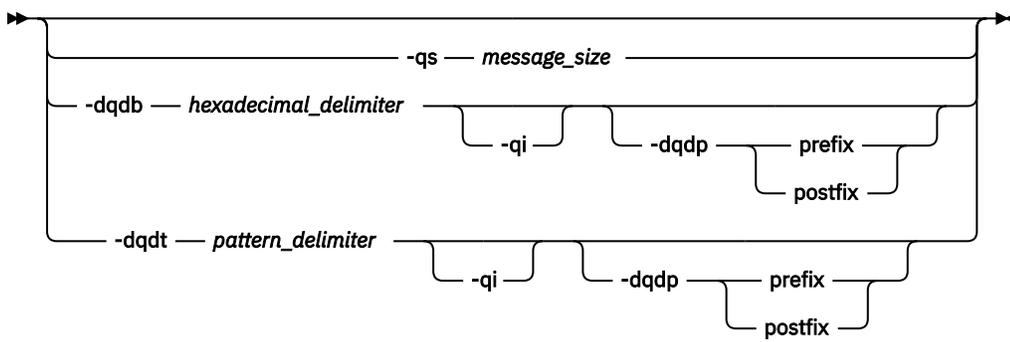
z/OS However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name.

Syntax

fteCreateTransfer



File splitting options



Parameters for MQ security



► source_specification ◄

Parameters for agent specification

-sa source_agent_name

Required. The name of the agent that the source files are transferred from.

z/OS If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

If you specify the **-td** parameter and the transfer definition file contains the source agent that you want to use for the transfer, do not specify the **-sa** parameter.

-sm source_agent_qmgr_name

Optional. The name of the queue manager that the source agent is connected to.

If you do not specify the **-sm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the source agent name. If the `agent.properties` file for the source agent cannot be found, the file transfer fails.

-da destination_agent_name

Required. The name of the agent that the files are transferred to.

If you specify the **-td** parameter and the transfer definition file contains the destination agent that you want to use for the transfer, do not specify the **-da** parameter.

-dm destination_agent_qmgr_name

Optional. The name of the queue manager that the destination agent is connected to.

If you do not specify the **-dm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the destination agent name. If the `agent.properties` file for the destination agent cannot be found, the file transfer fails.

Parameters for generating transfer templates

-gt transfer_template_file_path

Optional. Generates a transfer template XML message and writes this message to a file. If you specify this parameter, no transfer request is sent to Managed File Transfer. Instead, the contents of the transfer request message are written to the named XML document. You can then use this XML document to define the task for resource monitoring. See `fteCreateMonitor` command for information about how to create a resource monitor. If you do not specify this parameter, the default behavior takes place and an actual transfer request is carried out.

You must provide the full path and name of an XML output file as input for this parameter, for example `C:\templates\transfer_reports.xml`

z/OS On z/OS, you must store the transfer template document in a UNIX file on z/OS UNIX System Services. You cannot store transfer template documents in z/OS sequential files or PDS members.

IBM i On IBM i, you must store the transfer template document in the integrated file system.

The transfer template XML message that you create by using the **-gt** parameter is not the same as the transfer you create by using the **fteCreateTemplate** command, which means you cannot use the two different types of template interchangeably.

Note: If you want to generate a transfer template XML document by running the **fteCreateTransfer** command with the **-gt** parameter, and then provide that transfer template XML document as input to the **fteCreateTransfer** command using the **-td** parameter, you must ensure that the transfer template XML document was generated specifying those parameters that are mutually exclusive with the **-td** option.

The parameters mutually exclusive to the **-td** option are:

- **-dd** *destination_directory*
- *Source path*
- **-df** *destination_file*
- **-cs** *checksum*
- **-de** *destination_file_behavior*
- **-dq** *destination_queue*
- **-t** *transfer type*
- **-sd** *source_file_disposition*

For example, it is not possible to specify both the **-td** and **-t** parameters (indicating whether the transfer is a binary or text transfer) on the **fteCreateTransfer** command. This means that if you want to pass in a transfer template XML document to the command and specify that the transfer should be a text transfer, you should create the XML document by specifying the **-gt** and **-t** text parameters.

Parameters for scheduling transfers

-ss *schedule_start_time*

Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time by using the 24-hour clock:

```
yyyy-MM-ddThh:mm  
hh:mm
```

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

-tb

Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

admin

The start and end times used for the scheduled transfer are based on the time and date of the system used by the local administrator. This is the default value.

source

The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

UTC

The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

-oi

Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

minutes

hours

days

weeks

months

years

-of occurrence_frequency

Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

-oc occurrence_count

Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count is met, the scheduled transfer is deleted.

Specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

-es schedule_end_time

Optional. The time and date that a repeating scheduled transfer ends.

You must specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

Use one of the following formats to specify the end time and date. Specify the time by using the 24-hour clock:

```
yyyy-MM-ddThh:mm
```

```
hh:mm
```

Parameters for triggering transfers

-tr

Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

condition,namelist

where *condition* is one of the following values:

file=exist

A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

file!=exist

A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

filesize>=size

A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">"=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the same system as the source agent. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks.

You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

Note: To continually monitor a resource for a trigger condition to be true, you are strongly recommended to use [resource monitoring](#). You can create a resource monitor by using the `fteCreateMonitor` command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt,C:\export\B.txt
-tr filesize">"=1GB,C:\export\A.txt,C:\export\B.txt
-df C:\import\file1.doc C:\export\file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

The **-tr** parameter is not supported on protocol bridge agents.

-tl

Optional. Specifies whether trigger failures are written to the transfer log. Specify one of the following options:

yes

Transfer log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

no

No transfer log entries are created for failed triggered transfers.

Parameters for specifying transfer options

-jn job_name

Optional. A user-defined job name identifier that is added to the transfer log message when the transfer starts.

-md

Optional. Specifies the user-defined metadata that is passed to the exit points run by the agent. The **-md** parameter can take one or more name-value pairs that are separated by commas. Each name pair consists of *name=value*. You can use the **-md** parameter more than once in a command.

When the agent property `enableUserMetadataOptions` is set to a value of `true`, certain user-defined metadata keys provide more options to the transfer. For more information about the user-defined metadata keys that are currently supported, see [“enableUserMetadataOptions: Supported MFT user-defined metadata keys”](#) on page 548. When the `enableUserMetadataOptions` property is set to `true`, key names starting with `com.ibm.wmqfte.` are not supported for user-defined use.

-cs checksum

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

MD5

Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, Managed File Transfer computes MD5 checksums for all file transfers.

none

No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to `none` and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the `none` option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

-pr transfer_priority

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is the priority level of the source agent.

This value matches the message priority value of IBM MQ, see [Getting messages from a queue: priority](#) for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your IBM MQ message traffic to take priority.

-qmp boolean

Optional. Specifies whether the first message written to the destination queue by the transfer has IBM MQ message properties set. The valid options are as follows:

true

Sets message properties on the first message that is created by the transfer.

false

Does not set message properties on the first message that is created by the transfer. This is the default value.

You can specify the **-qmp** parameter only if you also specify the **-dq** parameter. For more information, see [“MQ message properties set by MFT on messages written to destination queues”](#) on page 651

-qs message_size

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The size of the messages is specified by the value of *message_size*. The format of *message_size* is *lengthunits*, where *length* is a positive integer value and *units* is one of the following values:

B

Bytes. The minimum value that is allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

K

This is equivalent to 1024 bytes.

M

This is equivalent to 1048576 bytes.

If the file is transferred in text mode, and is in a double-byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can specify the **-qs** parameter only if you also specify the **-dq** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-qi

Optional. Using this option includes the delimiter that is used to split the file into multiple messages in the messages. The delimiter is included at the beginning or at the end of the message, depending on the **-dqdp** parameter (which specifies prefix or postfix). By default the delimiter is not included in the messages.

You can specify the **-qi** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to create the file transfer. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options that are based on the default coordination queue manager is used.

-w timeout

Optional. Specifying the **-w** parameter causes the **fteCreateTransfer** command to wait for a response from the agent before returning. If you do not specify this parameter, the **fteCreateTransfer** command waits a maximum of five seconds to receive an acknowledgment from the source agent for the transfer that the agent has received the transfer request. If no acknowledgment is received during the five-second wait, the **fteCreateTransfer** command returns the following warning message:

```
BFGCL0253W: No acknowledgment to command from agent within timeout.
```

The *timeout* argument is optional. If you specify *timeout*, the **fteCreateTransfer** command waits for up to *timeout* seconds for the agent to respond. If the agent does not respond before the time limit is reached, the command produces a warning and ends with a return code of 2. If you do not specify a *timeout* value, or you specify a *timeout* value of -1, then the command waits until the agent responds.

V 9.0.1**-rt recovery_timeout**

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

-1

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

0

The agent stops the file transfer as soon as it enters recovery.

>0

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
-rt 21600
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this parameter is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a property to the [The agent.properties file](#).

Parameters for invoking programs

For more information about how you can start a program from Managed File Transfer, see “[Specifying programs to run with MFT](#)” on page 239. For examples of specifying a program to invoke using the parameters that are described here, see “[Examples of using fteCreateTransfer to start programs](#)” on page 818.

-presrc pre_source_call

Optional. Specifies a program to invoke at the source agent before the transfer starts. Use the following format for *pre_source_call*:

```
[type:]commandspec[, [retrycount][, [retrywait][, successsrc]]]
```

In this syntax, the variables are:

type

Optional. Valid values are **executable**, **antscript**, and **jcl**. The default value is **executable**.

 The **jcl** value is only applicable when targeted at an agent in a z/OS environment. In this case, the command refers to either a ZFS file, or a QSAM-readable dataset, or a member of a PDS. The contents should be JCL that can be submitted.

commandspec

Required. The command specification. Use one of the following formats:

- Type **executable**: *command*[(*arg1*, *arg2*, ...)]

If arguments contain variable substitutions, like `${FilePath}` or `${FileName}`, the variables are substituted with the first item in the transfer request.

For example, if a transfer request consists of files "reports01.csv, reports02.csv, reports03.csv", and the destination directory is "/output", the following transfer request:

```
fteCreateTransfer -sa 1 -da 2 -presrc "executable:archive(${FileName})"
-dd TargetDir "${FilePath}" -gt task.xml
```

is replaced with

```
fteCreateTransfer -sa 1 -da 2 -presrc "executable:archive(reports01.csv)"
-dd TargetDir "/ouptut" -gt task.xml
```

- Type **antscript**: *command*[(*name1=var1* | *target1*, *name2=var2* | *target2*, ...)]
- Type **jcl**: *command*

where:

command

Required. The name of the program to call.

The **jcl** value is only applicable when targeted at an agent in a z/OS environment.

Arguments in brackets ([]) are optional and syntax depends on command type. Parentheses, commas (,), and backslash (\) characters that are within the command or parameters must be escaped with a back slash (\) character.

retrycount

Optional. The number of times to retry calling the program if the program does not return a successful return code. Default value is 0.

retrywait

Optional. The time to wait, in seconds, before trying the program invocation again. Default value is 0 (no wait between retries).

successrc

Optional. Expression that is used to determine when the program invocation successfully runs. This expression can be composed of one or more expressions. Combine these expressions with a vertical bar character (|) to represent Boolean OR, or an ampersand (&) character to represent Boolean AND. Each expression is of the following form:

```
[>|<|!]value
```

where

>

Optional. A greater than test of the *value*.

<

Optional. A less than test of the *value*.

!

Optional. A not equal to test of the *value*.

value

Required. A valid integer.

-predst *pre_destination_call*

Optional. Specifies a program to invoke at the destination agent before the transfer starts. *pre_destination_call* has the same format as *pre_source_call*.

-postsrc *post_source_call*

Optional. Specifies a program to invoke at the source agent after the transfer has completed. *post_source_call* has the same format as *pre_source_call*.

-postdst *post_destination_call*

Optional. Specifies a program to invoke at the destination agent after the transfer has completed. *post_destination_call* has the same format as *pre_source_call*.

Parameters for specifying the destination

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, and **-dp** parameters is required. You cannot specify more than one of these parameters in a transfer request; they are mutually exclusive.

-td *transfer_definition_file*

Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer. Alternatively, the name of the XML document that contains a managed transfer request (which might have been generated by the **-gt** parameter). If you specify the **-td** parameter and also specify any other parameters on the command line, these other parameters override the corresponding value from the transfer definition file.

The **ftCreateTransfer** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

z/OS On z/OS, you must store the transfer definition file in a UNIX file on z/OS UNIX System Services. You cannot store transfer definition files in z/OS sequential files or PDS members.

IBM i On IBM i, you must store the transfer definition file in the integrated file system.

For more information, see [Using transfer definition files](#).

-df destination_file

Optional. The name of the destination file.

If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format *connect_direct_node_name:file_path*. The Connect:Direct bridge agent accepts only file paths that are specified in this format. **z/OS** If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of `overwrite`.

Note the following information:

- If the destination agent is a protocol bridge agent and you want to specify an endpoint for a file, use the following format:

```
protocol_server:file_path
```

where *protocol_server* is the name of the protocol server (which is optional) and where *file_path* is the path to the file on the protocol server system. If you do not specify a protocol server, the default protocol server is used.

- If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-df** parameter in a transfer.
- **z/OS** When the destination agent is on z/OS, if the file specified starts with `//`, it is assumed to be a partitioned z/OS data set.

-dd destination_directory

Optional. The name of the directory the file is transferred to. Specify a valid directory name on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format *connect_direct_node_name:directory_path*. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of `overwrite`.

Note the following information:

- If the destination agent is a protocol bridge agent and you want to specify a directory at a particular endpoint, use the following format:

```
protocol_server:directory_path
```

where *protocol_server* is the name of the protocol server (which is optional) and where *directory_path* is the path to the directory on the protocol server system. If you do not specify a protocol server, the default protocol server is used.

- If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-dd** parameter in a transfer.
- **z/OS** When the destination agent is on z/OS, if the file specified starts with `//`, it is assumed to be a z/OS partitioned data set.

-ds destination_sequential_data_set

z/OS only. Optional. The name of the sequential data set or PDS member that files are transferred into. Specify a sequential data set name or a partitioned data set member. For information about transferring data sets, see [“Guidelines for transferring files”](#) on page 604.

The syntax for the data set name is as follows:

```
//data_set_name{;attribute(value);...;attribute(value)}
```

or

```
//pds_data_set_name(member_name){;attribute(value);...;attribute(value)}
```

That is, a data set name specifier prefixed with // and optionally followed by a number of attributes that are separated by semicolons.

For example:

```
// 'TEST.FILE.NAME' ;DSNTYPE(PDS);RECFM(F,B);BLKSIZE(800);LRECL(80);CYL;SPACE(2,2)
```

If the data set is located at a Connect:Direct node, you must prefix the data set name with the node name. For example:

```
CD_NODE1:// 'OBJECT.LIB' ;RECFM(F,B);BLKSIZE(800);LRECL(80)
```

If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite. For more information about data set transfers to or from Connect:Direct nodes, see [“Transferring data sets to and from Connect:Direct nodes”](#) on page 609.

For transfers that only involve Managed File Transfer agents, if the data set name part is enclosed by single quotation mark characters, it specifies a fully qualified data set name. If the data set name is not enclosed by single quotation mark characters, the system adds the default high-level qualifier for the destination agent (either the value for the transferRootHLQ agent property or the user ID that the agent runs under, if you have not set transferRootHLQ).

Note:  However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name. This is the case even if the data set name is enclosed by single quotation mark characters.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes that are passed in the transfer definition. If no attributes are specified, attributes are set to the same as the source data set or to the default values when the source is a file. The attributes of an existing tape data set are ignored.

The data set attributes are used either to create a data set or to ensure that an existing data set is compatible. The specification of data set attributes is in a form suitable for BPXWDYN (see [Requesting dynamic allocation](#) for more information). When the agent is to create a destination data set, the following BPXWDYN attributes are automatically specified: DSN(*data_set_name*) NEW CATALOG MSG(*numeric_file_descriptor*). The value of *numeric_file_descriptor* is generated by Managed File Transfer. For a data set to data set transfer, the attributes of RECFM, LRECL, and BLKSIZE from the source are selected for a new destination data set. The SPACE setting for a new destination data set is not set by Managed File Transfer and system defaults are used. Therefore, you are recommended to specify the SPACE attribute when a new data set is to be created. You can use the **bpxwdynAllocAdditionalProperties** property in the agent .properties file to set BPXWDYN options that apply to all transfers. For more information, see [“The MFT agent.properties file”](#) on page 526.

z/OS Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalProperties** property in the `agent.properties` file. For a list of these properties, see [“BPXWDYN properties you must not use with MFT” on page 616](#).

The **-ds** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-ds** parameter in a transfer. Using the **-ds** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

z/OS **-dp destination_partitioned_data_set**

z/OS only. Optional. The name of the destination PDS that files are transferred into. Specify a partitioned data set name. If a PDS is created as a result of the transfer, this PDS is created as a PDSE by default. You can override the default by specifying `DSNTYPE=PDS`.

The syntax for the PDS data set name is as follows:

```
//pds_data_set_name{;attribute;..;attribute}
```

The syntax for the data set name is the same as described for the **-ds** (*destination_sequential_data_set*) parameter. All the syntax details for specifying data sets that are located on Connect:Direct nodes also apply to the **-dp** parameter. If the destination agent is a Connect:Direct bridge agent, you must also specify the **-de** parameter with a value of `overwrite`.

The **-dp** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dp** parameter in a transfer. Using the **-dp** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

-dq destination_queue

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, by using the format `QUEUE@QUEUEMANAGER`. If you do not specify a queue manager name the destination agent queue manager name is used. You must specify a valid queue name that exists on the queue manager.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dq** parameter in a transfer. Using the **-dq** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

-dqp persistent

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

true

Writes persistent messages to the destination queue. This is the default value.

false

Writes non-persistent messages to the destination queue.

qdef

The persistence value is taken from the `DefPersistence` attribute of the destination queue.

You can specify the **-dqp** parameter only if you also specify the **-dq** parameter.

-dqdb hexadecimal_delimiter

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST_MSG_IN_GROUP flag set. The format for specifying a hexadecimal byte as a delimiter is xNN, where N is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: x3e , x20 , x20 , xbf.

You can specify the **-dqdb** parameter only if you also specify the **-dq** parameter and the transfer is in binary mode. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdt pattern

Optional. Specifies the Java regular expression to use when splitting a text file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST_MSG_IN_GROUP flag set. The format for specifying a regular expression as a delimiter is a regular expression that is enclosed in parentheses, (*regular_expression*), or enclosed in double quotation marks, "*regular_expression*". For more information, see [“Regular expressions used by MFT” on page 633](#).

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior by editing the **maxDelimiterMatchLength** agent property. For more information, see [“Advanced agent properties” on page 527](#).

You can specify the **-dqdt** parameter only if you also specify the **-dq** parameter and the value text for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdp position

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can specify the **-dqdp** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

prefix

The delimiters are expected at the beginning of each line.

postfix

The delimiters are expected at the end of each line. This is the default option.

-de destination_file_behavior

Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

error

Reports an error and the file is not transferred. This is the default value.

overwrite

Overwrites the existing destination file.

If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify destination file exists behavior in the transfer definition file.

-t transfer type

Optional. Specifies the type of file transfer: binary mode or text mode.

binary

The data in the file is transferred without any conversion. This is the default value.

text

The code page and end-of-line characters of the file are converted. You can specify which code page and line ending to use for the conversion with the **-sce**, **-dce** or **-dle** parameters. If you do not specify the **-sce**, **-dce** or **-dle** parameters, the exact conversions performed depend on the operating system of the source agent and destination agent.

► **z/OS** For example, a file that is transferred from Windows to z/OS has its code page converted from ASCII to EBCDIC. When a file is converted from ASCII to EBCDIC, the end-of-line characters are converted from ASCII carriage return (CR) and line feed (LF) character pairs to an EBCDIC new line (NL) character.

► **z/OS** For more information about how z/OS data sets are transferred, see [Transferring files and data sets between z/OS and distributed systems](#) and [Transferring between data sets](#).

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

-dce destination_character_encoding

Optional. Specifies which character encoding to use to write the file at the destination. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent. For a list of available code pages, see the topic “Available code pages for MFT” on page 657.

noswaplfnl

By default Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to the EBCDIC LF 0x25 character. However, this can sometimes result in a mapping that is not what you want. Use noswaplfnl to override this behavior.

-dle destination_line_ending

Optional. Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only and so you must also specify the **-t text** parameter. The valid options are:

LF

Line feed. This is the default for the following platforms:

- ► **UNIX** UNIX platforms
- ► **z/OS** z/OS UNIX System Services files

When you use the standard EBCDIC code pages that are supplied with Managed File Transfer for EBCDIC files, the end-of-line characters are mapped to a NL character (0x15) and not to a LF character (0x25).

CRLF

Carriage return followed by line feed. ► **Windows** This is the default for Windows.

► **z/OS** If the destination of the transfer is a z/OS data set, this option is ignored.

-dtr

Optional. Specifies that destination records longer than the LRECL data set attribute are truncated. If this parameter is not specified, the records are wrapped. This parameter is valid only for text mode transfers where the destination is a data set.

-dfa attributes

Optional. When transferring to an IBM MQ 8.0 Managed File Transfer agent running on an 4690, this parameter is used to specify a semicolon separated list of file attributes that are associated with the destination files in the transfer. The **-dfa** parameter can be specified with or without a value. For example, without a value:

```
-dfa ATTRIBUTE1;ATTRIBUTE2
```

For example, with a value:

```
-dfa ATTRIBUTE1(VALUE);ATTRIBUTE2(VALUE)
```

For example, one attribute with a value and one without:

```
-dfa ATTRIBUTE1;ATTRIBUTE2(VALUE)
```

You can use the **-dfa** parameter more than once in a command.

For more information about file attributes on 4690, see [File distribution attributes](#) in IBM Documentation.

Parameters for security

-mquserid (*userID*)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (*password*)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

Parameters for specifying the source

-sd *source_file_disposition*

Optional. Specifies the action that is taken on a source file in file-to-file or file-to-message transfers when that source file is successfully transferred to its destination. The valid options are as follows:

leave

The source files are left unchanged. This is the default value.

delete

The source files are deleted from the source system after the source files are successfully transferred.

Note: For message-to-file transfers, the messages on the source queue are always deleted once they have been successfully transferred. This means that if the **-sd** parameter is set to `leave` for a message-to-file transfer, the value is ignored.

 On z/OS, if the source is a tape data set and you specify the `delete` option, the tape is remounted to delete the data set. This behavior is because of the behavior of the system environment.

If the source is a queue and you specify the `leave` option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the `delete` option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

- If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the `delete` option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [“Submitting a user-defined Connect:Direct process from a file transfer request”](#) on page 229.
- If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

-r

Optional. Recursively transfer files in subdirectories when *source_specification* contains wildcard characters. When Managed File Transfer is presented with a wildcard character as a

source_specification, any subdirectories that match the wildcard character are transferred only if you specify the **-r** parameter. When *source_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how Managed File Transfer handles wildcard characters, see [Using wildcard characters](#)

If you specify the **-r** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

-sce *source_character_encoding*

Optional. Specifies which character encoding to use to read the source file when performing character conversion. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent, because the conversion is performed on the destination system. For a list of available code pages, see the topic [“Available code pages for MFT”](#) on page 657.

noswaplfnl

By default Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to the EBCDIC LF 0x25 character. However, this can sometimes result in a mapping that is not what you want. Use noswaplfnl to override this behavior.

z/OS

-skeep

Optional. Specifies that trailing spaces are kept on source records that are read from a fixed-length-format record-oriented file (for example, a z/OS data set) as part of a text mode transfer. If you do not specify this parameter, trailing spaces are stripped from source records.

z/OS

-srdb *delimiter*

Optional. For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file. You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. Separate multiple bytes with commas. For example:

```
-srdb x0A
```

or

```
-srdb x0D,x0A
```

You must configure the transfer in binary mode.

z/OS

-srdp *position*

Optional. Specifies the position to insert source record delimiters. You can specify the **-srdp** parameter only if you also specify the **-srdb** parameter.

Specify one of the following options:

prefix

The delimiters are inserted at the start of each record.

postfix

The delimiters are inserted at the end of each record. This is the default option.

-sq

Optional. Specifies that the source of a transfer is a queue.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-sq** parameter in a transfer. Using the **-sq** parameter prevents the

transfer I/O user exits from being invoked for the source and means that the standard Managed File Transfer I/O is used instead.

-sqgi

Optional. Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

You can specify the **-sqgi** parameter only if you also specify the **-sq** parameter.

-sqdt text_delimiter

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `-sqdt \u007d\n`.

The text delimiter is encoded to binary format by using the source encoding of the transfer. Each message is read in binary format. The encoded delimiter is prepended or appended in binary format to the message (as specified by the **-sqdp** parameter) and the result is transferred in binary format to the destination agent. If the source agent code page includes shift-in and shift-out states, the agent assumes that each message is in the shift-out state at the end of the message. At the destination agent the binary data is converted in the same way as a file to file text transfer.

You can specify the **-sqdt** parameter only if you also specify the **-sq** parameter and the value `text` for the **-t** parameter.

-sqdb hexadecimal_delimiter

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by `x`. Multiple bytes must be comma-separated. For example, `-sqdb x08,xA4`.

You can specify the **-sqdb** parameter only if you also specify the **-sq** parameter. You cannot specify the **-sqdb** parameter if you also specify the value `text` for the **-t** parameter.

-sqdp position

Optional. Specifies the position of insertion of source text and binary delimiters. You can specify the **-sqdp** parameter only if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

prefix

The delimiters are inserted at the start of each message

postfix

The delimiters are inserted at the end of each message. This is the default option.

-sqwt wait_time

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to appear on the queue
- If the **-sqgi** parameter was specified, for a complete group to appear on the queue

If neither of these conditions is met within the time that is specified by `wait_time`, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.

For information about using the **-sqwt** parameter, see [“Guidance for specifying a wait time on a message-to-file transfer”](#) on page 657.

You can specify the **-sqwt** parameter only if you also specify the **-sq** parameter.

source_specification

One or more file specifications that determine the source, or sources, for the file transfer.

Required if you specify one of the **-df**, **-dd**, **-dp**, **-dq**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source_specification*.

- If you have not specified the **-sq** parameter, *source_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters, see [“Using wildcard characters with MFT” on page 627](#). You can escape asterisks that are part of the file specification by using two asterisk characters (**) in the file specification.

You can specify multiple source file specifications separated by the space character. However, if you specify multiple source specifications for the **-df** or **-ds** parameters and also specify **-de overwrite**, the destination will contain only the data for the source file that you specified last. If you do not specify **-de overwrite** the transfer can only be partially successful. If the destination file did not previously exist, it will contain the data for the source file that you specified first.

To transfer files that contain spaces in their file names, for example a b.txt to file c d.txt, place double quotation marks around the file names that contain spaces. Specify the following text as part of the **fteCreateTransfer** command:

```
-df "c d.txt" "a b.txt"
```

Each file specification must be in one of the following categories:

File names

The name of a file, expressed in the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.

Directories

The name of a directory, expressed in the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.

For example, to copy the contents of DIR1 to DIR2 only, specify `fteCreateTransfer . . . -dd DIR2 DIR1/*`

Sequential data set

The name of a sequential data set or partitioned data set member. Denote data sets by preceding the data set name with two forward slash characters (//).

If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

Partitioned data set

The name of a partitioned data set. Denote data set names by preceding the data set name with two forward slash characters (//).

If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

File name or directory at a Connect:Direct node

(Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (:), and a file or directory path on the system that is hosting the Connect:Direct node. For example, `connect_direct_node_name:file_path`.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

Note: Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

File name or directory on a protocol file server

The name of a protocol file server, a colon character (:), and a file or directory path on the protocol server system. For example, `protocol_server:file_path`.

If you do not specify a protocol server, the default protocol server is used.

- If you specify the **-sq** parameter, *source_specification* is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

```
QUEUE_NAME
```

The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

- **z/OS** If the source agent is on z/OS, source files that start with // are assumed to be z/OS partitioned data sets.

Other parameters

-? or -h

Optional. Displays command syntax.

Examples

In this basic example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2 on the same system and renamed to `transferredfile.txt`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, the files `originalfile.txt` and `originalfile2.txt` are transferred from AGENT1 to AGENT2 on the same system, to the directory `C:\import`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import C:\export\originalfile.txt  
C:\export\originalfile2.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1's system to AGENT2's system. The file transfer is scheduled to take place at 09:00 based on the system time of the source agent's system and occurs every two hours four times:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE  
-tb source -ss 09:00 -oi hours -of 2 -oc 4  
-df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2, on condition that the file `A.txt` exists on AGENT1:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE  
-tr file=exist,C:\export\A.txt -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

z/OS In this example, the file `originalfile.txt` is transferred from AGENT1's system to a data set `//'USERID.TRANS.FILE.TXT'` on AGENT2's system. Text mode is selected to convert data from ASCII to EBCDIC.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2  
-ds "//TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);  
SPACE(5,1)" C:\export\originalfile.txt
```

z/OS In this example, a member of a fully qualified data set on AGENT1's system is transferred to a file on AGENT2's system. Text mode is selected to convert the file from EBCDIC to the default code page of AGENT2's system.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2 -df /tmp/IEEUJV.txt "'/SYS1.SAMPLIB(IEEUJV)'"
```

In this example, a file that is called `file.bin` on agent AGENT1 is transferred to a destination file called `file.bin` on the protocol file server `accountshost.ibm.com` by using the destination agent BRIDGE1.

```
fteCreateTransfer -sa AGENT1 -da BRIDGE1 -df accountshost.ibm.com:/tmp/file.bin /tmp/file.bin
```

In this example, a wildcard is used without quotation marks. All files in AGENT1's current working directory that end in `.txt` are transferred to directory `C:\import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import *.txt
```

In this example, a wildcard is used with double quotation marks. All files in AGENT1's transfer root directory that end in `.txt` are transferred to directory `C:\import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import "*.txt"
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.
2	Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time specified.
20	Command completed with partial success and some files were transferred.
21	The queue manager that the fteCreateTransfer command was connected to was stopped before the transfer result was determined.
40	Failed. None of the files specified were transferred.
41	The transfer was canceled.
42	The transfer did not take place because the transfer was conditional and the required condition was not met.
43	The transfer request message was malformed.
44	The source agent did not have sufficient capacity to carry out the transfer.
45	The destination agent did not have sufficient capacity to carry out the transfer.
46	The number of files that are being transferred exceeded the source agent's limit.
47	The number of files transferred exceeds the destination agent's limit.

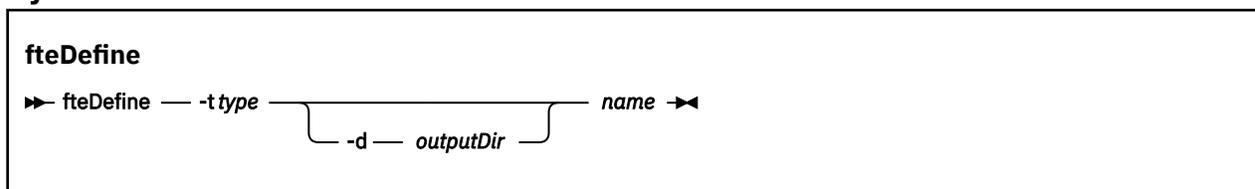
fteDefine (generate configuration scripts)

Use the **fteDefine** command to generate the configuration scripts necessary to define the specified Agent Queue Manager objects.

Purpose

You would expect to use the **fteDefine** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, configuring the queues for an agent on a queue manager to be accessed over a client connection.

Syntax



Parameters

-t *type*

Required. The type of object to be defined. The options for type are agent.

-d *outputDir*

Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

name

Required. One or more names of the objects to be defined. To specify names for more than one object, separate them with a space. For example, *name1 name2 . . .*

-? or -h

Optional. Displays command syntax.

Examples

In this example, the **fteDefine** command is specified with the **-t agent** parameter and a single agent name. The output is written to a file.

```
fteDefine -t agent EXAMPLE.AGENT >EXAMPLE.AGENT_create.mqsc
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to create the necessary agent queues:

```
$ fteDefine -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
DEFINE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT) +
  DEFPRTY(0) +
  DEFSOPT(SHARED) +
  GET(ENABLED) +
  MAXDEPTH(5000) +
  MAXMSGL(4194304) +
  MSGDLVSQ(PRIORITY) +
  PUT(ENABLED) +
  RETINTVL(999999999) +
  SHARE +
  NOTRIGGER +
  USAGE(NORMAL) +
  REPLACE
DEFINE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT) +
  DEFPRTY(0) +
  DEFSOPT(SHARED) +
  GET(ENABLED) +
  MAXDEPTH(5000) +
  MAXMSGL(4194304) +
```

```
MSGDLVSQ(PRIORITY) +
PUT(ENABLED) +
RETINTVL(999999999) +
SHARE +
NOTRIGGER +
USAGE(NORMAL) +
REPLACE
...
etc.
```

In this example, the **fteDefine** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```
$ fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
EXAMPLE.AGENT.1.
The file can be found here: '/tmp/EXAMPLE.AGENT.1_create.mqsc'.
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
EXAMPLE.AGENT.2.
The file can be found here: '/tmp/EXAMPLE.AGENT.2_create.mqsc'.
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
EXAMPLE.AGENT.3.
The file can be found here: '/tmp/EXAMPLE.AGENT.3_create.mqsc'.
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

“[fteDelete \(generate scripts to remove configuration\)](#)” on page 440

Use the **fteDelete** command to generate the configuration scripts necessary to remove the specified Agent Queue Manager objects.

fteDelete (generate scripts to remove configuration)

Use the **fteDelete** command to generate the configuration scripts necessary to remove the specified Agent Queue Manager objects.

Purpose

You would expect to use the **fteDelete** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, removing the queues for a remote client agent on a local queue manager.

Syntax

fteDelete

```
► fteDelete -t type -d outputDir name ◄
```

Parameters

-t type

Required. The type of object to be delete. The options for type are agent.

-d outputDir

Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

name

Required. One or more names of the objects to be delete. To specify names for more than one object, separate them with a space. For example, *name1 name2 . . .*

-? or -h

Optional. Displays command syntax.

Examples

In this example, the **fteDelete** command is specified with the **-t agent** parameter and a single agent name. The output is written to a file.

```
fteDelete -t agent EXAMPLE.AGENT >EXAMPLE.AGENT_delete.mqsc
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to delete the agent queues:

```
$ fteDelete -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
...
etc.
```

In this example, the **fteDelete** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```
$ fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.1.
The file can be found here: '/tmp/EXAMPLE.AGENT.1_delete.mqsc'.
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.2.
The file can be found here: '/tmp/EXAMPLE.AGENT.2_delete.mqsc'.
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.3.
The file can be found here: '/tmp/EXAMPLE.AGENT.3_delete.mqsc'.
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

“[fteDefine \(generate configuration scripts\)](#)” on page 439

Use the **fteDefine** command to generate the configuration scripts necessary to define the specified Agent Queue Manager objects.

fteDeleteAgent

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

Purpose

Stop the agent with the [fteStopAgent](#) command before running the **fteDeleteAgent** command.

If you have configured your agent to run as a Windows service, running the **fteDeleteAgent** command deletes the service definition.

V 9.0.4 From IBM MQ 9.0.4, any resource monitor and scheduled transfers are removed when the agent is deleted.

Only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive an error message and the command will not run.

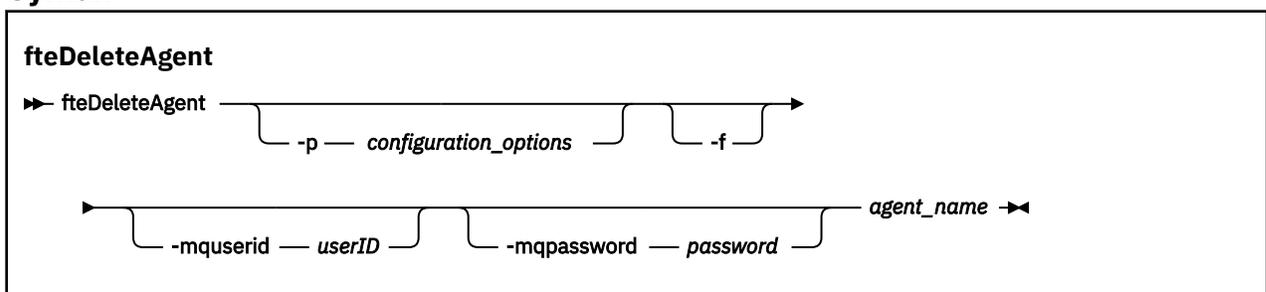
The **fteDeleteAgent** command provides you with the MQSC commands that you must run against your agent's queue manager to clear and delete the agent's system queues. These queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

The **fteCreateAgent** command also provides these commands in a file in the following location:

```
MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/agent_name_delete.mqsc
```

Syntax



Parameters

-p (configuration_options)

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agent configuration you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the `installation.properties` file are used. See [“MFT configuration options on Multiplatforms” on page 56](#) for more information.

-f

Optional. Forces the command to deregister the agent from the coordination queue manager even if the agent's configuration files cannot be found. Because information about the agent's queue manager is not available in this situation, the command will connect directly to the coordination queue manager instead of using the agent queue manager as it normally would.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the agent queue manager, unless the force **-f** parameter is present. If the **-f** parameter is present, it specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the agent queue manager, unless the force **-f** parameter is present. If the **-f** parameter is present, it specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

agent_name

Required. The name of the agent that you want to delete.

-? or -h

Optional. Displays command syntax.

Example

In this example, AGENT3 and its configuration on coordination queue manager QM_COORD1 are deleted:

```
fteDeleteAgent -p QM_COORD1 AGENT3
```

This example command outputs the following MQSC commands to delete the agent's three queues:

```
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.DATA.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.DATA.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.STATE.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.STATE.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
```

```
CLEAR QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“fteStopAgent” on page 508](#)

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

[“fteCleanAgent: clean up an MFT Agent” on page 366](#)

Use the **fteCleanAgent** command to clean up the queues that a Managed File Transfer Agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

[“fteCreateAgent \(create an MFT agent\)” on page 371](#)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

[“fteStartAgent” on page 505](#)

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

fteDeleteLogger

Use the **fteDeleteLogger** command to delete a Managed File Transfer logger and its configuration. Existing log files associated with the logger can either be retained or deleted.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Purpose

Stop the logger with the **fteStopLogger** command before running the **fteDeleteLogger** command.

If you have configured your logger to run as a Windows service, running the **fteDeleteLogger** command deletes the service definition.

The logger configuration directory contains a MQSC script to delete the queues and the subscription for the logger. These queues are as follows:

- SYSTEM.FTE.LOG.CMD.*logger_name*
- SYSTEM.FTE.LOG.RJCT.*logger_name*

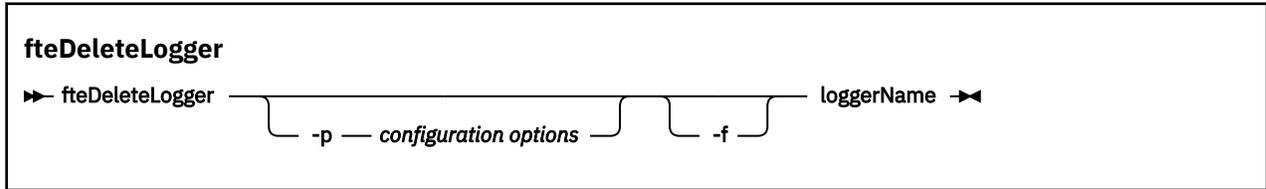
The subscription name is as follows:

- SYSTEM.FTE.AUTO.SUB.*logger_name*

The MQSC script can be found at

MQ_DATA_PATH\mqft\config\coordination_qmgr\loggers*logger_name**logger_name_delete*.mqsc

Syntax



Parameters

-p (*configuration_options*)

Optional. Determines the set of configuration options that is used to start the stand-alone database logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f

Optional. Forces the removal of any log files created by this logger. If this parameter is omitted, any log files created by the logger will be retained, and must be manually removed when they are no longer required.

logger_name

Required. The name of the logger that you want to delete.

-? or -h

Optional. Displays command syntax.

Example

In this example, a logger called `logger1` is deleted. The **-f** parameter has been specified, which causes the logger's log files to be removed as well as the logger's configuration files.

```
fteDeleteLogger -f logger1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“fteCreateLogger \(create an MFT file or database logger\)” on page 389](#)

Use the **fteCreateLogger** command to create a Managed File Transfer file or database logger.

[“fteStartLogger” on page 507](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteStopLogger” on page 510](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an

IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

fteDeleteMonitor: delete an MFT resource monitor

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

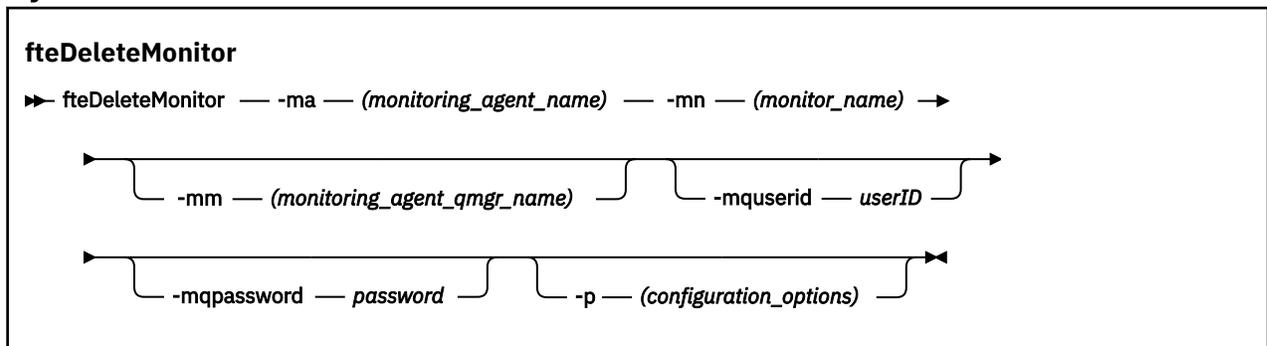
Purpose

Use the **fteDeleteMonitor** command to stop monitoring a resource and remove the monitor's definition from the monitoring agent. When you run this command, no more polls of the resource occur and no further tasks are started.

You can run the **fteDeleteMonitor** command from any system that can connect to the IBM MQ network and subsequently route to the agent's queue manager. Specifically for the command to run, you must have installed a Managed File Transfer component (either Service or Agent) on this system and you must have configured this system's Managed File Transfer to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Syntax



Parameters

-ma (monitoring_agent_name)

Required. The name of the agent that performs the resource monitoring. This monitoring agent must also have been the source agent for the file transfer that you wanted to trigger.

-mn (monitor_name)

Required. The name that you assigned to this resource monitor. You can delete a resource monitor and then create a new monitor with the same name.

-mm (monitoring_agent_qmgr_name)

Optional. The name of the monitoring agent's queue manager. Because the monitoring agent and the source agent of the transfer the monitor triggered must be same, this queue manager is also your source agent's queue manager.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-p (configuration_options)

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-? or -h

Optional. Displays command syntax.

Example

In this example, the resource monitor MONITOR1 with a monitoring (and file transfer source agent) AGENT1 is deleted:

```
fteDeleteMonitor -ma AGENT1 -mm QM_JUPITER -mn MONITOR1
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Monitoring MFT resources” on page 162](#)

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

Related reference

[“fteCreateMonitor: create an MFT resource monitor” on page 394](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

[“fteListMonitors: list MFT resource monitors” on page 455](#)

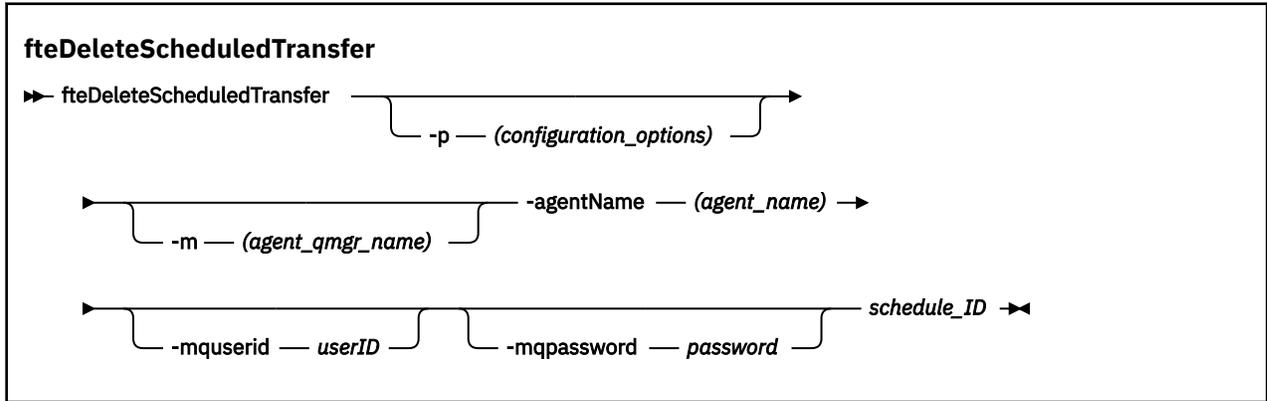
Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

fteDeleteScheduledTransfer**Purpose**

Use the **fteDeleteScheduledTransfer** command to delete a Managed File Transfer scheduled transfer that you have previously created either using the command line or the IBM MQ Explorer.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See [“MFT configuration options on Multiplatforms” on page 56](#) for more information.

Syntax



Parameters

-p (configuration_options)

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which scheduled transfer you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

-m (agent_qmgr_name)

Optional. The name of the queue manager that the source agent is connected to. If you do not specify this parameter, the agent's queue manager is determined from the configuration options in use.

-agentName (agent_name)

Required. The name of the source agent that you want to delete the scheduled transfer from.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

schedule_ID

Required. The ID of the scheduled transfer that you want to delete.

You can find the schedule ID by running the [fteListScheduledTransfers](#) command against the name of the source agent.

-? or -h

Optional. Displays command syntax.

Example

In this example, a scheduled transfer on source agent AGENT2 with the ID 27 is deleted:

```
fteDeleteScheduledTransfer -agentName AGENT2 27
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Creating a scheduled file transfer” on page 155](#)

You can schedule a new file transfer either from the IBM MQ Explorer Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

Related reference

[“fteListScheduledTransfers” on page 458](#)

Use the **fteListScheduledTransfers** command to list all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.

fteDeleteTemplates

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

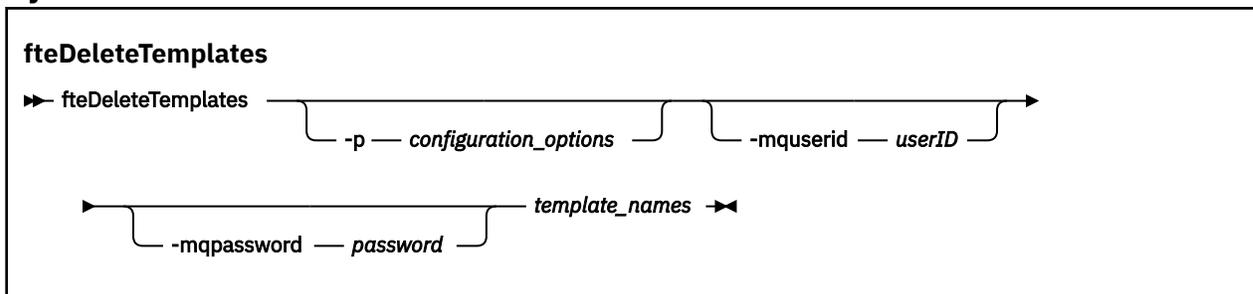
Purpose

The **fteDeleteTemplates** command removes one or more file transfer templates from a coordination queue manager. When you run this command a request is passed to the IBM MQ system to remove the templates from the coordination queue manager so that the templates are no longer available to the IBM MQ Explorer or the command line. The templates you are deleting might continue to be accessed for a brief interval after the command completes until the IBM MQ system actions the request.

You can run the **fteDeleteTemplates** command from any system that can connect to the IBM MQ network and subsequently route to the coordination queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured this system's Managed File Transfer to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Syntax



Parameters

-p (configuration_options)

Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

(template_names)

Required. Specify one or more template names that you want to delete. Specify the name as displayed by the **fteListTemplates** command.

-? or -h

Optional. Displays command syntax.

Example

In this example, the template STANDBY is deleted:

```
fteDeleteTemplates STANDBY
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Working with file transfer templates” on page 186](#)

You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the IBM MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your Managed File Transfer network.

[“Creating a file transfer template using IBM MQ Explorer” on page 188](#)

You can create a file transfer template from IBM MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference

[“fteCreateTemplate: create new file transfer template” on page 401](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn (template_name)** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[“fteListTemplates” on page 459](#)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

fteDisplayVersion

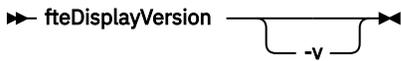
Use the **fteDisplayVersion** command to display the version of Managed File Transfer that you have installed.

Purpose

You might be asked to run the **fteDisplayVersion** command by an IBM Service Representative to help with problem determination.

Syntax

fteDisplayVersion



Parameters

-v

Optional. Displays a verbose amount of information about the product version.

The precise details that are displayed when you specify the **-v** parameter might vary between product releases. You are not recommended to rely on specific information being available in the output from the `fteDisplayVersion -v` command.

 On z/OS, **-v** displays the value of the **productId** property, if the product Id has been specified.

-? or -h

Optional. Displays command syntax.

Example with no parameters specified

In this example, the **fteDisplayVersion** command is specified with no parameters.

```
fteDisplayVersion
```

The output from this command is the product version level. For example, this is the output for IBM MQ 9.0:

```
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2016. ALL RIGHTS RESERVED
IBM MQ Components:
Name:      IBM MQ Managed File Transfer
Version:   9.0.0.0
```

Example with **-v** parameter specified

In this example, the **fteDisplayVersion** command is specified with the **-v** parameter.

```
fteDisplayVersion -v
```

The output from this command includes more detailed information about the product version. For example:

```
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2016. ALL RIGHTS RESERVED
IBM MQ Components:
Name:      IBM MQ Managed File Transfer
Version:   9.0.0.0
Level:     p900-L160512.4
Platform:  Windows 10 (10.0)
Architecture: amd64
JVM:      JRE 1.8.0 Windows 10 amd64-64 Compressed References 20160427_301573 (JIT enabled, AOT
enabled)
          J9VM - R28_Java8_SR3_20160427_1620_B301573
          JIT  - tr.r14.java.green_20160329_114288
          GC   - R28_Java8_SR3_20160427_1620_B301573_CMPRSS
          J9CL - 20160427_301573
Product:   C:\Program Files\IBM\MQ_1
Configuration: C:\ProgramData\IBM\MQ\mqjft

Name:      IBM MQ JMS Provider
```

```
Version: 9.0.0.0
Level: p900-L160512.4

Name: IBM MQLight Service for Bluemix JMS Provider
Version:
Level: p900-L160512.4

Name: Common Services for Java Platform, Standard Edition
Version: 9.0.0.0
Level: p900-L160512.4

Name: Java Message Service Client
Version: 9.0.0.0
Level: p900-L160512.4

Name: IBM MQ classes for Java Message Service
Version: 9.0.0.0
Level: p900-L160512.4

Name: IBM MQ classes for Java
Version: 9.0.0.0
Level: p900-L160512.4
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

fteListAgents: list the MFT agents for a coordination queue manager

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

Purpose

You can run the **fteListAgents** command from any system that can connect to the coordination queue manager. The following details for each agent are directed to the standard output device (STDOUT):

- Agent name
- Agent queue manager
- If the agent is a protocol bridge agent, the agent name is appended with `bridge` .
- If the agent is a Connect:Direct bridge agent, the agent name is appended with `(Connect:Direct bridge)`
- Agent status

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [“The MFT coordination.properties file” on page 517](#).

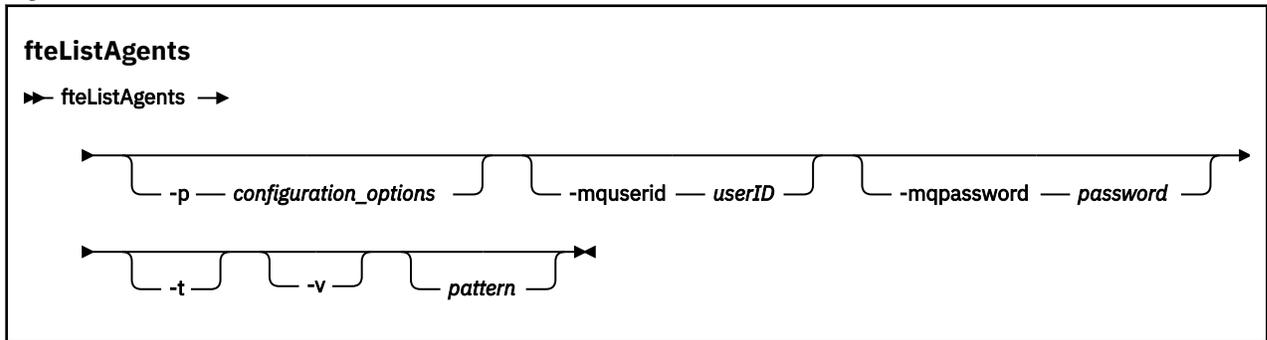
Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [“MFT configuration options on Multiplatforms” on page 56](#).

If an agent is not listed by the **fteListAgents** command, use the diagnosis flowchart in the following topic to locate and fix the problem: [If your agent is not listed by the fteListAgents command](#).

Agent status information

The agent status information produced by this command is generated from the status messages that the agent publishes to the SYSTEM.FTE topic. These messages are described in the topic [“MFT agent status message format” on page 709](#). The status information produced by the **fteListAgents** command gives the agent status at the time when the last status message was published. The frequency of these status messages depends on the value of the `agentStatusPublishRateLimit` property. For more details about this property, see the topic [“The MFT agent.properties file” on page 526](#).

Syntax



Parameters

-p (configuration_options)

Optional. This parameter determines the set of configuration options that is used to issue the request to list agents. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-v

Optional. Specifies verbose mode. Verbose mode generates additional output for each agent, including the current number of transfers in the form `Source/Destination`, where `Source` is the current number of source transfers and `Destination` is the current number of destination transfers.

The current transfer information is obtained from the agent status publication, which is described in the following topic: [“MFT agent status message format” on page 709](#). As a result, this transfer information is only accurate to within the setting for the `agentStatusPublishRateLimit` agent property value (which defaults to 30 seconds).

-t

Optional. Specifies terse mode. At IBM MQ 9.0.2, the output includes the **Status Age** column by default. If you do not want to see the **Status Age** information, you can issue the command with the **-t** parameter to hide the column. For more information, see [What to do if the `fteListAgents` command shows an agent status of UNKNOWN](#).

pattern

Optional. The pattern to use to filter the list of Managed File Transfer agents. This pattern is matched against the agent name. Asterisk (*) characters are interpreted as wildcards, that match any value, including zero characters.

On UNIX and Linux systems, you must escape special characters like the asterisk (*) and the number sign (#) with quotation marks (') or double quotation marks (") if you want them to be handled as literals. If you do not escape these characters, they are interpreted according to their meaning on the specific UNIX or Linux system.

If you do not specify this parameter, all agents registered with the coordination queue manager are listed.

-? or -h

Optional. Displays command syntax.

Example

In this example, all of the agents registered on the queue manager detailed in the configuration options with names beginning with B are listed:

```
fteListAgents "B*"
```

In this example, agents that are registered with the coordination queue manager QM_EUROPE (the non-default coordination queue manager) are listed in verbose mode:

```
fteListAgents -p QM_EUROPE -v
```

The output from this command is as follows:

Agent Name:	Queue Manager Name:	Transfers: (Source/Destination)	Status:
BERLIN	QM_BERLIN	7/0	RUNNING
LONDON	QM_LONDON	0/0	RUNNING
MADRID	QM_MADRID	0/1	UNREACHABLE

For a list of the possible agent status values and their meanings, see the topic [“MFT agent status values”](#) on page 599.

In this example, all agents that are registered with the coordination queue manager and that have names beginning with BRIDGE are listed in verbose mode:

```
fteListAgents -v "BRIDGE*"
```

The output from this command is as follows:

```
C:\Program Files\IBM\WMQFTE\bin>fteListAgents -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Name:                               Queue Manager Name:   Transfers:   Status:
                                           (Source/Destination)
BRIDGE_FTP ( bridge )                     QM_JUPITER           0/0         STOPPED
BRIDGE_CD1 (Connect:Direct bridge)        QM_JUPITER           0/0         STOPPED
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Listing MFT agents”](#) on page 203

You can list the Managed File Transfer agents registered with a particular queue manager using the command line or the IBM MQ Explorer.

Related reference

[“MFT agent status values”](#) on page 599

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

[“fteShowAgentDetails”](#) on page 495

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

fteListMonitors: list MFT resource monitors

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

Purpose

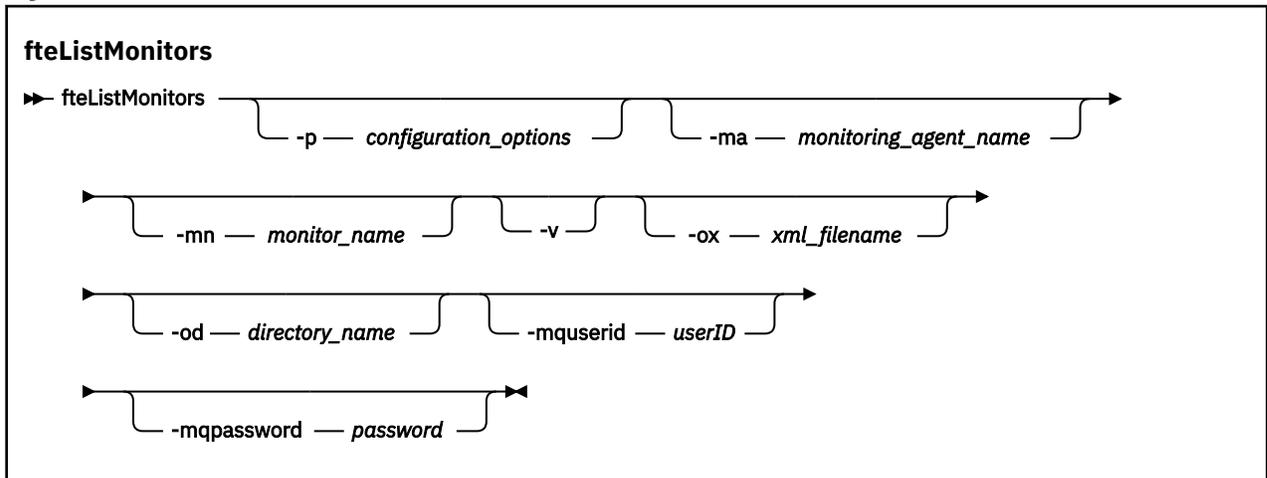
The **fteListMonitors** command lists existing resource monitors. You can filter the command output by specifying an agent name and a resource monitor name.

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [“The MFT coordination.properties file” on page 517](#).

You can use the **-ox** parameter to export a resource monitor to an XML file. For more information on how to use this XML file, see [“fteCreateMonitor: create an MFT resource monitor” on page 394](#).

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [Configuration options](#).

Syntax



Parameters

-p (configuration_options)

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-ma (monitoring_agent_name)

Optional. Filters resource monitors by agent name by using the pattern that you provide as input. Asterisk (*) characters are interpreted as wildcards that match zero or more characters. If you do not specify the **-ma** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

-mn (monitor_name)

Optional. Filters resource monitors by monitor name by using the pattern that you provide as input. Asterisk (*) characters are interpreted as wildcards that match zero or more characters. If you do

not specify the **-mn** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password. The password is not displayed.

-v

Optional. Generates verbose output that includes additional information about the status of the monitor, including whether the monitor is started or stopped, the directory resource path that is being monitored and the trigger conditions.

-ox (xml_filename)

Optional. You must specify this parameter in combination with the **-ma** and **-mn** parameters. Exports the resource monitor to an XML file that can then be used by the **fteCreateMonitor** command and the **-ix** parameter.

V 9.0.5 The **-ox** parameter must not be combined with the **-od** parameter.

V 9.0.5 **-od (directory_name)**

Optional. Exports multiple resource monitor definitions to the specified directory. Each resource monitor definition is saved to a separate XML file with a name in the format *agent_name.monitor_name.xml*. You must specify a valid target directory for the XML files, otherwise an error message is displayed. This parameter must not be combined with the **-ox** parameter.

-? or -h

Optional. Displays command syntax.

Example: list resource monitors

In this example, all resource monitors associated with the monitoring agent (and source agent for the file transfers associated with the monitor) AGENT1 are listed:

```
fteListMonitors -ma AGENT1
```

Example: export one resource monitor to an XML file

In this example, a single resource monitor, MONITOR1, on AGENT1 is exported to the XML file filename1.xml by specifying an XML file name with the **-ox** parameter:

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
```

Example: export one resource monitor to a specified directory

V 9.0.5

In this example, a single resource monitor, MONITOR1, on AGENT1 is exported to the directory that is specified by the **-od** parameter. Except for the difference in the XML file name format, this example is similar to using the **-ox** parameter.

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -od /usr/mft/resmonbackup
```

Examples: export a batch of resource monitors to an XML file in a specified directory

V 9.0.5

In all of the following examples, the resource monitors are exported to the directory that is specified by the **-od** parameter. Each resource monitor definition is saved to separate XML file with a name in the format *agent name.monitor name.xml*.

In this example, all resource monitors are exported to the specified directory:

```
fteListMonitors -od /usr/mft/resmonbackup
```

In this example, all resource monitors on AGENT1 are exported to the specified directory:

```
fteListMonitors -ma AGENT1 -od /usr/mft/resmonbackup
```

You can use wildcard matching to define which resource monitors to export by using an asterisk character (*) when you specify a pattern to match to agent names, or monitor names, or both.

In this example, all resource monitors on AGENT1 with names that match the pattern MON* are exported to the specified directory:

```
fteListMonitors -ma AGENT1 -mn MON* -od /usr/mft/resmonbackup
```

In this example, all resource monitors on agents with names that match the pattern AGEN* are exported to the specified directory:

```
fteListMonitors -ma AGEN* -od /usr/mft/resmonbackup
```

In this example, all resource monitors with names that match the pattern MON* on agents with names that match the pattern AGEN* are exported to the specified directory:

```
fteListMonitors -ma AGEN* -mn MON* -od /usr/mft/resmonbackup
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Monitoring MFT resources” on page 162](#)

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

[“Backing up and restoring MFT resource monitors” on page 184](#)

You can back up the resource monitors that you want to have available for future use by exporting their definitions to an XML file that you can then import to create a new resource monitor from the backup.

Related reference

[“fteCreateMonitor: create an MFT resource monitor” on page 394](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

[“fteDeleteMonitor: delete an MFT resource monitor” on page 446](#)

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

fteListScheduledTransfers

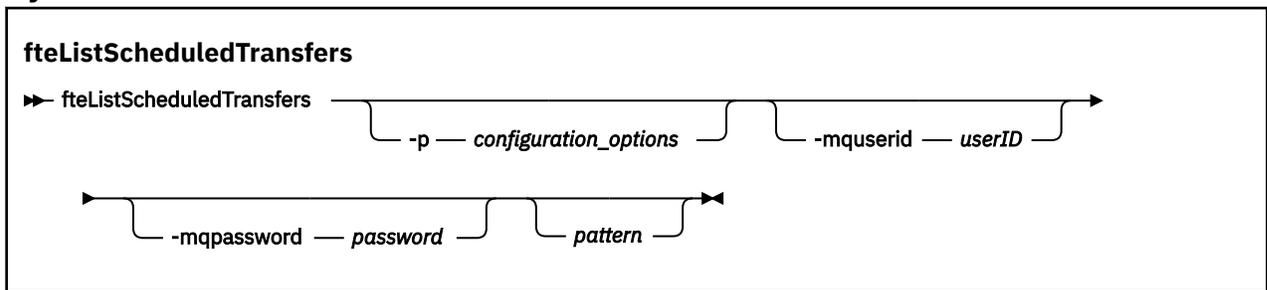
Use the **fteListScheduledTransfers** command to list all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.

Purpose

You can either list all scheduled transfers based on source agent names or based on the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See [“MFT configuration options on Multiplatforms” on page 56](#) for more information.

Syntax



Parameters

-p (configuration_options)

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agents you want to list scheduled transfers for. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

pattern

Optional. The pattern to use to filter the list of Managed File Transfer scheduled transfers. This pattern is matched against the source agent name. Asterisk (*) characters are interpreted as wildcards that match zero or more characters.

If you do not specify this parameter, all of the scheduled transfers registered with the coordination queue manager are listed by default.

-? or -h

Optional. Displays command syntax.

Example

In this example, all of the scheduled transfers with source agents that match the pattern *2 are listed:

```
fteListScheduledTransfers "*2"
```

This example command produces the following output. The schedule start time and next transfer time are displayed in Coordinated Universal Time (UTC):

```
Schedule Identifier:      1
Source Agent Name:       AGENT2
Source File Name:        C:/export/Test/workspace/A.exe
Conversion Type:         binary
Destination File Name:   C:/import/Test/workspace/B001.zzx
Destination Agent Name:  AGENT1
Schedule Start Time:     2008-10-23T16:08+0100
Next Transfer:           2008-10-23T16:08+0100
Schedule Time Base:     source
Repeat Interval:         minutes
Repeat Frequency:        1
Repeat Count:            30
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Creating a scheduled file transfer” on page 155](#)

You can schedule a new file transfer either from the IBM MQ Explorer Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

Related reference

[“fteDeleteScheduledTransfer” on page 447](#)

fteListTemplates

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

Purpose

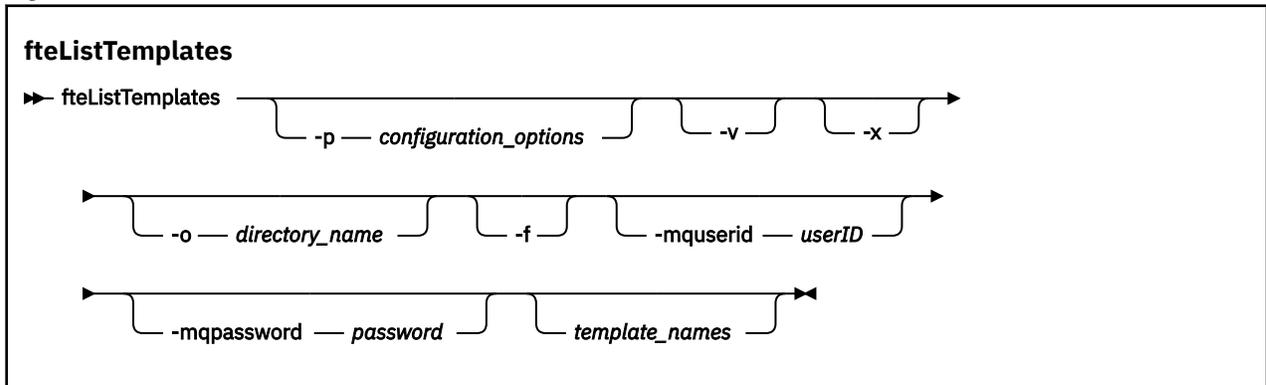
This command lists either all template names or a filtered selection of template names. The output format of the list can be any of the following:

- Template names only (default behavior)
- Template names with a summary of the templates (verbose mode)
- Complete XML message describing the templates (**-x** and **-o** parameters)

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [“The MFT coordination.properties file” on page 517](#).

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Syntax



Parameters

-p

Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-v

Optional. Specifies verbose mode and provides a short summary of each matching template. This parameter is ignored if you have also specified the **-x** parameter.

The **-v** parameter includes a summary of each template. For example:

```
Template Name: STANDBY
Source Agent Name: AGENT1
Source QMgr: QM_JUPITER
Destination Agent Name: AGENT2
Destination QMgr: QM_NEPTUNE
Transfer Priority: 0
Transfer file specification
File Item Details
  Mode: binary
  Checksum: MD5
  Source File:
    C:\payroll_reports\*.xls
  Recursive: false
  Disposition: leave
  Destination File:
    C:\payroll_backup\*.xls
  Type: file
  Exist: error
```

If you do not specify the **-v** parameter, the default output mode is to list the matching templates names.

-x

Optional. Provides an XML-formatted message for each matching template. This parameter is ignored unless you also specify the **-o** parameter.

-o (*directory_name*)

Optional. Sends the XML formatted-message to files in the named directory. One file for each template is created and each file has the same name as the template with an `.xml` suffix. This parameter is ignored unless you also specify the **-x** parameter.

-f

Optional. Forces any existing output file to be overwritten. This parameter is ignored unless you also specify the **-o** parameter. If you do not specify **-f** but you do specify the name of an existing output file, the default behavior is to report an error and continue.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

(template_names)

Optional. A list of one or more template names to be listed. A template name can include an asterisk as a wildcard that matches zero or more characters. Depending on your operating system, you might need to enclose any template names that include wildcard character in quotation marks (" ") or single quotation marks (' ') to avoid shell expansion. Shell expansion can cause unexpected behavior.

If you do not specify anything for *template_names*, the default is to list all templates.

-? or -h

Optional. Displays command syntax.

Example

In this example, all the templates with names starting with ST are listed:

```
fteListTemplates "ST*"
```

This example creates the template STANDBY as an XML-formatted message to the file STANDBY.xml in the current directory:

```
fteListTemplates -x -o . STANDBY
```

This command creates the following output in STANDBY.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <transferTemplate id="1864c1dd-ba02-4b34-bda9-dc6862448418" version="3.00">
  <name>STANDBY</name>
  <sourceAgentName>AGENT1</sourceAgentName>
  <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
  <sourceAgentQMgrHost>null</sourceAgentQMgrHost>
  <sourceAgentQMgrPort>-1</sourceAgentQMgrPort>
  <sourceAgentQMgrChannel>null</sourceAgentQMgrChannel>
  <destinationAgentName>AGENT2</destinationAgentName>
  <destinationAgentQMgr>QM_NEPTUNE</destinationAgentQMgr>
- <fileSpecs>
  - <item checksumMethod="MD5" mode="binary">
    - <source disposition="leave" recursive="false">
      <file>C:\payroll_reports\*.xls</file>
    </source>
    - <destination exist="error" type="file">
      <file>C:\payroll_backup\*.xls</file>
    </destination>
  </item>
</fileSpecs>
  <priority>0</priority>
</transferTemplate>
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later

If you want to migrate an existing agent and its associated configuration from IBM WebSphere MQ File Transfer Edition 7.0 to IBM WebSphere MQ 7.5 or later, use the **fteMigrateAgent** command to migrate. This command can be used to migrate a standard agent, a Connect:Direct agent, or a protocol bridge agent. The command can also be used to migrate multiple agents in a single request.

V 9.0.0 From IBM MQ 9.0, Managed File Transfer does not support web agents. If you attempt to use the **fteMigrateAgent** command to migrate a web agent from an earlier release to IBM MQ 9.0, an error message is displayed to explain that the migration of a web agent is not supported.

Note: If you are migrating from IBM WebSphere MQ File Transfer Edition 7.0 or later, and want to continue using the FTE_CONFIG environment variable, you can do so without changing the FTE_CONFIG value. You can perform a standard migrate, but BFG_DATA must not be set, and FTE_CONFIG must be set as used in IBM WebSphere MQ 7.0.

Important: **ULW** On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

z/OS On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- **V 9.0.0.1** Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
- **V 9.0.0.1** Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

If your agent is configured to run as a Windows service, use the **fteModifyAgent** command to reconfigure the agent so that it is no longer a Windows service. After the migration is complete, use the **fteModifyAgent** command again to configure the new agent to be a Windows service. Alternatively, if you include the **-f** parameter, the command completes but produces a warning.

Before you can run the **fteMigrateAgent** command, you must stop the agent you want to migrate using the **fteStopAgent** command.

If you run the command with the **-f** parameter, only the information about the agent is refreshed. If a required file is missing, the command fails.

Specifically, the following properties files, XML files, and directory associated with the agent are migrated:

Name of the file migrated by the fteMigrateAgent command for each agent	Information
wmqfte.properties	The wmqfte.properties file is renamed to installation.properties in IBM WebSphere MQ 7.5 or later.
command.properties	
coordination.properties	
coordination_queue_manager.mqsc	
agent_name_create.mqsc	

Table 37. Agent files migrated by the `fteMigrateAgent` command (continued)

Name of the file migrated by the <code>fteMigrateAgent</code> command for each agent	Information
<code>agent_name_delete.mqsc</code>	
exits directory	The command copies all files in the exits directory.
Applies to standard agents only:	
<code>UserSandboxes.xml</code>	
Applies to Connect:Direct bridge agents only:	
<code>ConnectDirectCredentials.xml</code>	
<code>ConnectDirectNodeProperties.xml</code>	
<code>ConnectDirectProcessDefinitions.xml</code>	
Applies to protocol bridge agents only:	
<code>ProtocolBridgeCredentials.xml</code>	
<code>ProtocolBridgeProperties.xml</code>	This file exists on IBM WebSphere MQ File Transfer Edition 7.0.4 or later only.

The **`fteMigrateAgent`** command migrates the files for the installation, coordination, and command queue managers and copies them to IBM WebSphere MQ 7.5 or later if the files do not already exist on IBM WebSphere MQ 7.5 or later. If the files already exist, they are not copied as part of the command.

Syntax

<p>fteMigrateAgent</p> <p>► <code>fteMigrateAgent</code> — <code>-agentName</code> — <code>agent_name</code> — <code>-config</code> — <code>configuration_directory</code> →</p> <p> ► <code>-credentialPath</code> — <code>credentials_path</code> — <code>-f</code> — <code>-p</code> — <code>configuration_options</code> —►</p>

Parameters

`-agentName` `agent_name`

Required. The name of the agent that you want to migrate to IBM WebSphere MQ 7.5 or later.

`-config` `configuration_directory`

Required. The path to the configuration directory for the installation that you are migrating the agent from. For example, `C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config`

`-credentialPath` `credentials_path`

Required. Defines the location to migrate the credential information to. This parameter can either be a directory path where existing credential files are present or a new location to receive a new credential file. For z/OS platforms this can be a pre-existing partitioned data set extended (PDSE), either with existing members to be updated, or without existing members to include a new member for these credentials.

Note: If a PDSE is used, it must be variable blocked.

-f

Optional. Forces the agent to migrate even if some of the configuration files that are typically migrated conflict with the existing configuration. For example, if there is a mismatch between the properties files on Managed File Transfer and the properties files on IBM WebSphere MQ 7.5, or later, specifying the **-f** parameter means that this mismatch is ignored.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to locate the configuration to migrate. Use the name of a set of configuration options as the value of the **-p** parameter. By convention this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used. For more information, see [“MFT configuration options on Multiplatforms”](#) on page 56.

-? or -h

Optional. Displays command syntax.

Examples

In this example, AGENT3 and its configuration in `/var/ibm/WMQFTE/config` is migrated to IBM WebSphere MQ 7.5 or later:

```
fteMigrateAgent -agentName AGENT3 -config /var/ibm/WMQFTE/config -credentialPath /home/user1/AGENT3
```

In this example, all agents and their configurations in `C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config` are migrated to IBM WebSphere MQ 7.5 or later. The Windows file path is enclosed in double quotation marks (" "). The **-f** parameter is specified to force migration and ignore any property file mismatches:

```
fteMigrateAgent -agentName "*" -config "C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config" -credentialPath "C:\Documents and Settings\user1\AGENT3" -p "configurationOption" -f
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

For more information about return codes, see [“Return codes for MFT”](#) on page 322.

fteMigrateConfigurationOptions: migrate an FTE 7.0 configuration to MQ 7.5 or later

The **fteMigrateConfigurationOptions** command migrates a set of configuration options from IBM WebSphere MQ File Transfer Edition 7.0 and copies them to IBM WebSphere MQ 7.5 or later, provided that the files do not already exist on the target version. If the files already exist, a message is output and the command does not continue.

Note: If you are migrating from IBM WebSphere MQ File Transfer Edition 7.0, and want to continue using the `FTE_CONFIG` environment variable, you can do so without changing the `FTE_CONFIG` value. You can perform a standard migrate, but `BFG_DATA` must not be set, and `FTE_CONFIG` must be set as used in IBM WebSphere MQ File Transfer Edition 7.0.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message `BFGCL0502E: You are not authorized to perform the requested operation.` and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- **V9.0.0.1** Be a member of the group named in the BFG_GROUP_NAME environment variable (if one is named).
- **V9.0.0.1** Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

Syntax

fteMigrateConfigurationOptions

```

▶▶ fteMigrateConfigurationOptions — -config — configuration_directory — -credentialPath →
    ▶ — credentials_path — -configurationOptionsName — configuration_options_name ▶▶

```

Parameters

-config (*configuration_directory*)

Required. The path to the configuration directory for the installation that you are migrating from. For example, C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config

-credentialPath (*credentials_path*)

Required. Defines the location to migrate the credential information to. This parameter can either be a directory path where existing credential files are present or a new location to receive a new credential file.

This parameter is used for migrating password properties for the SSL/TLS key store and trust store properties that are present in the `agent.properties`, `coordination.properties`, and `command.properties` files from a version of the product earlier than IBM WebSphere MQ 7.5 to 7.5 or later.

Before IBM WebSphere MQ 7.5, the password property, for example **coordinationSslTrustStorePassword**, used to be present in the `coordination.properties` file, but was moved to the `MQMFTCcredentials.xml` file in IBM WebSphere MQ 7.5. To move the password property from the `coordination.properties` file to the `MQMFTCcredentials.xml` file, the **-credentialsPath** is used. When this option is used the **fteMigrateConfigurationOptions** command looks in the `coordination.properties` file for "coordinationSslTrustStorePassword" and, if it is present, migrates the property to the `MQMFTCcredentials.xml` file.

z/OS For z/OS platforms this can be a pre-existing partitioned data set extended (PDSE), either with existing members to be updated, or without existing members to include a new member for these credentials.

Note: If a PDSE is used, it must be variable block.

-configurationOptionsName (*configuration_options_name*)

Required. The name of the set of configuration options that you want to migrate. You can migrate multiple sets of configuration options by using the asterisk character (*) to represent zero or more characters. You can use an asterisk with a string. For example, to migrate all sets of configuration options with names beginning with IBM, use this parameter as follows: `-configurationOptionsName IBM*`.

Examples

In this example, all configurations in the directory C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config are migrated. The directory path is enclosed in double quotation marks:

```
fteMigrateConfigurationOptions -config "C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config" -credentialPath "C:\Documents and Settings\user1\configurationoptions" -configurationOptionsName *
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“Migration considerations for MFT” on page 12](#)

If you are planning to migrate Managed File Transfer, review the following information.

[“fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later” on page 462](#)

If you want to migrate an existing agent and its associated configuration from IBM WebSphere MQ File Transfer Edition 7.0 to IBM WebSphere MQ 7.5 or later, use the **fteMigrateAgent** command to migrate. This command can be used to migrate a standard agent, a Connect:Direct agent, or a protocol bridge agent. The command can also be used to migrate multiple agents in a single request.

[“fteMigrateLogger: migrate an FTE 7.0 database logger to MQ 7.5 or later” on page 466](#)

If you want to migrate the configuration of an existing stand-alone database logger from IBM WebSphere MQ File Transfer Edition 7.0.1 or later to IBM WebSphere MQ 7.5 or later, use the **fteMigrateLogger** command.

fteMigrateLogger: migrate an FTE 7.0 database logger to MQ 7.5 or later

If you want to migrate the configuration of an existing stand-alone database logger from IBM WebSphere MQ File Transfer Edition 7.0.1 or later to IBM WebSphere MQ 7.5 or later, use the **fteMigrateLogger** command.

You cannot use this command to migrate a Java EE database logger: instead use the information in [Migrating a Java EE database logger](#).

Note: If you are migrating from IBM WebSphere MQ 7.0 or later, and want to continue using the FTE_CONFIG environment variable, you can do so without changing the FTE_CONFIG value. You can perform a standard migrate, but BFG_DATA must not be set, and FTE_CONFIG must be set as used in IBM WebSphere MQ 7.0.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

- Be a member of the mqm group (if the mqm group is defined on the system).
-  Be a member of the group named v in the BFG_GROUP_NAME environment variable (if one is named).
-  Have no value set in the BFG_GROUP_NAME environment variable when the command is run.

If you have configured a stand-alone database logger to run as a Windows service, you cannot migrate that logger's configuration using the **fteMigrateLogger** command. If you run the **fteMigrateLogger** command on a logger configured to run as a Windows service, the command produces an error and does not continue. Alternatively, if you include the -f parameter, the command completes but produces a warning.

Before you run the **fteMigrateLogger** command, stop the database logger whose configuration you want to migrate on IBM WebSphere MQ File Transfer Edition 7.0.

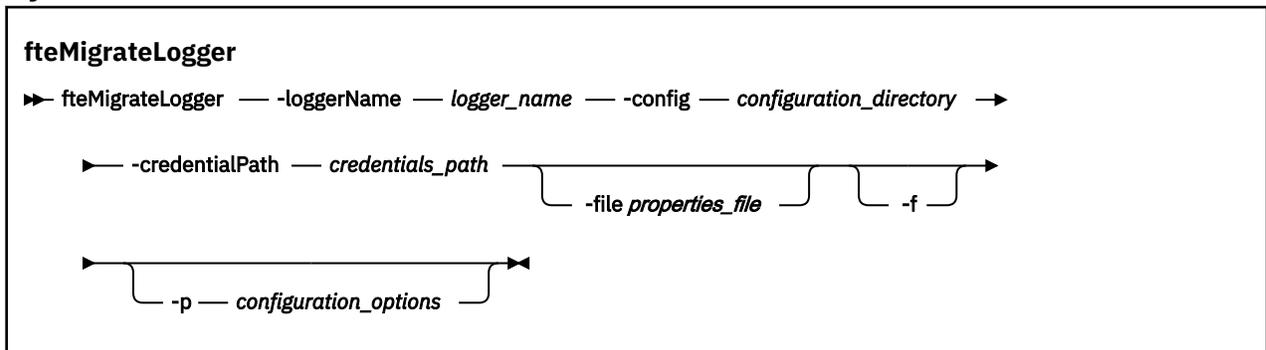
If you run the command with the -f parameter, only the information about the logger is refreshed. If a required file is missing, the command fails. Specifically, the following properties files and .mqsc file associated with the logger configuration are migrated:

Table 38. Files migrated by the fteMigrateLogger command

Name of the file migrated by the fteMigrateLogger command	Information
wmqfte.properties	The wmqfte.properties file is based on installation.properties in IBM WebSphere MQ 7.5 or later
command.properties	
coordination.properties	
coordination_queue_manager.mqsc	
databaselogger.properties or other properties file specified using the -file parameter	The databaselogger.properties is used to create the logger.properties file in IBM WebSphere MQ 7.5 or later.

The **fteMigrateLogger** command migrates the files for the installation, coordination, and command queue managers and copies them to IBM WebSphere MQ 7.5 or later provided that the files do not already exist on the target version. If the files already exist, they are not copied as part of the command.

Syntax



Parameters

-loggerName *logger_name*

Required. The name that you want to give to the migrated logger configuration in IBM WebSphere MQ 7.5 or later. For more information about logger names, which are new for IBM WebSphere MQ 7.5 see [logger_name](#) parameter.

-config *configuration_directory*

Required. The path to the configuration directory for the installation that the logger configuration is being migrated from.

-credentialPath *credentials_path*

Required. Defines the location to migrate the credential information to. This parameter can either be a directory path where existing credential files are present or a new location to receive a new credential file. For z/OS platforms this can be a pre-existing partitioned data set extended (PDSE), either with existing members to be updated, or without existing members to include a new member for these credentials.

Note: If a PDSE is used, it must be variable block.

-file *properties_file*

Optional. Specifies the database logger properties file to migrate. This parameter is required only if the properties file does not use the following default name and path: *configuration_directory/coordination_qmgr_name/databaselogger.properties*

-f

Optional. Forces migration even if some of the configuration files that are typically migrated conflict with the existing configuration. For example, if there is a mismatch between the database logger properties files on IBM WebSphere MQ File Transfer Edition and the properties files on IBM WebSphere MQ 7.5 or later, specifying the **-f** parameter means this mismatch is ignored.

-p *configuration_options*

Optional. This parameter determines the set of configuration options that is used to locate the logger configuration to migrate. Use the name of a set of configuration options as the value of the **-p** parameter. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used. For more information, see [“MFT configuration options on Multiplatforms”](#) on page 56.

-? or -h

Optional. Displays command syntax.

Example

In this example, the configuration of a stand-alone database logger located in `/var/ibm/WMQFTE/config` is migrated to IBM WebSphere MQ 7.5 and is named FTELOGGER1:

```
fteMigrateLogger -loggerName FTELOGGER1 -config /var/ibm/WMQFTE/config  
-credentialPath /home/user1/FTELOGGER1
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

For more information about return codes, see [“Return codes for MFT”](#) on page 322.

After running the `fteMigrateLogger` command

To verify the migration, after you have successfully run the **fteMigrateLogger** command, start the database logger whose configuration you have migrated on IBM WebSphere MQ 7.5 or later, using the [“fteStartLogger”](#) on page 507 command.

Related reference

[“Migration considerations for MFT”](#) on page 12

If you are planning to migrate Managed File Transfer, review the following information.

[“fteMigrateAgent: migrate an FTE 7.0 agent to MQ 7.5 or later”](#) on page 462

If you want to migrate an existing agent and its associated configuration from IBM WebSphere MQ File Transfer Edition 7.0 to IBM WebSphere MQ 7.5 or later, use the **fteMigrateAgent** command to migrate.

This command can be used to migrate a standard agent, a Connect:Direct agent, or a protocol bridge agent. The command can also be used to migrate multiple agents in a single request.

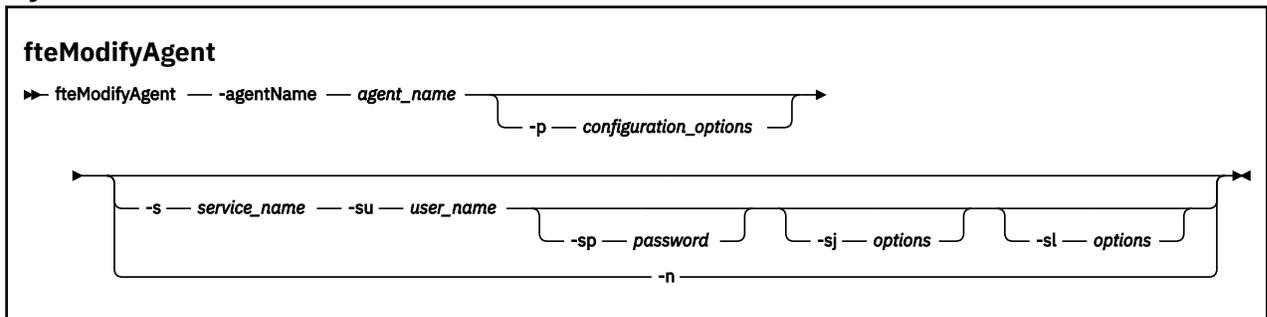
“[fteMigrateConfigurationOptions: migrate an FTE 7.0 configuration to MQ 7.5 or later](#)” on page 464

The **fteMigrateConfigurationOptions** command migrates a set of configuration options from IBM WebSphere MQ File Transfer Edition 7.0 and copies them to IBM WebSphere MQ 7.5 or later, provided that the files do not already exist on the target version. If the files already exist, a message is output and the command does not continue.

Windows **fteModifyAgent (run an MFT agent as a Windows service)**

The **fteModifyAgent** command modifies an existing agent so that it can be run as a Windows service. This command is only available on Windows and must be run by a user who is an IBM MQ administrator and a member of the mqm group.

Syntax



Parameters

-agentName agent_name

Required. The name of the agent you want to modify.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to modify the agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-s service_name

Optional. Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

-su user_name

Optional. When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service should run. To run the agent using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Guidance for running an MFT agent or logger as a Windows service](#).

This parameter is required when **-s** is specified.

-sp password

Optional. This parameter is only valid when **-s** is specified.

-sj options

Optional. When the agent is started as a Windows service, this parameter defines a list of options in the form of **-D** or **-X** that will be passed to the Java Virtual Machine (JVM). The options are separated using the number sign (**#**) or semicolon (**;**) character. If you need to embed any **#** or **;** characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

For more information about the way in which the **fteModifyAgent** command handles the validation of updates to the JVM options see [Guidance for updating agent or logger JVM options](#).

-sl options

Optional. Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

-n

Optional. Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the agent is configured as a normal Windows process.

-? or -h

Optional. Displays command syntax.

Example

In this example, AGENT1 is modified to run as a Windows service:

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

In this example, AGENT1 is modified to remove the Windows service:

```
fteModifyAgent -agentName AGENT1
```

You must stop the agent you want to modify, using the [fteStopAgent](#) command, before you can run the [fteModifyAgent](#) command.

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[Guidance for running an MFT agent or logger as a Windows service](#)

Related tasks

[Starting an MFT agent as a Windows service](#)

Related reference

“[fteCreateAgent \(create an MFT agent\)](#)” on page 371

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

“[fteModifyLogger \(run an MFT logger as a Windows service\)](#)” on page 471

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

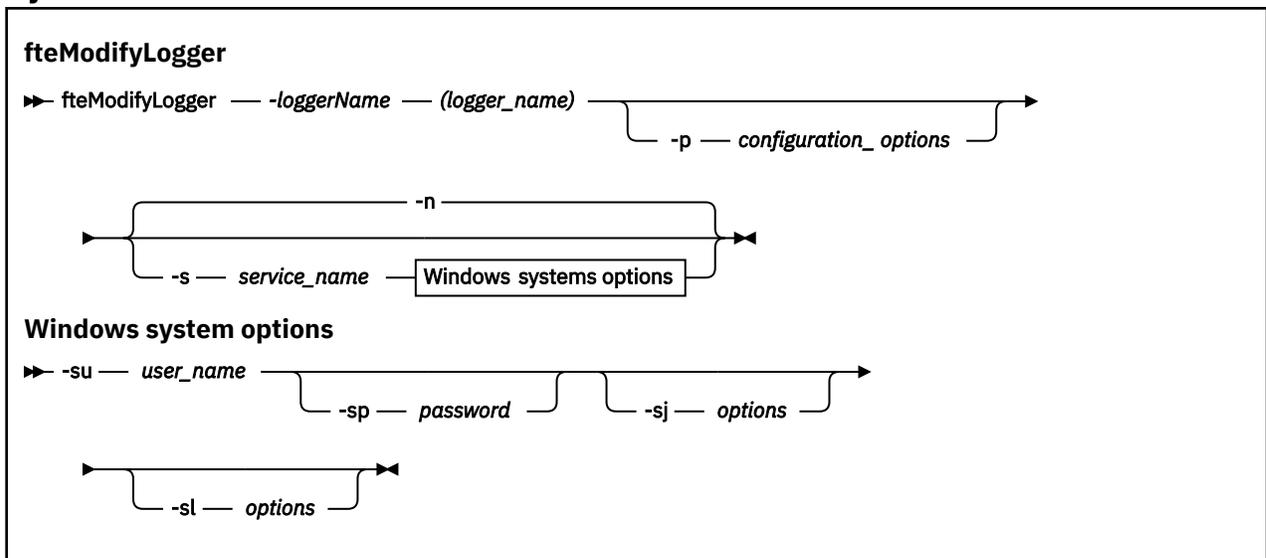
Windows **fteModifyLogger (run an MFT logger as a Windows service)**

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

Purpose

A stand-alone logger, whether for a file or for a database, is shown as "Managed File Transfer logger for property set *logger_name@logger_qm*" in the **Name** column of the **Services** application. The value of *logger_qm* is the name of the command queue manager of the logger.

Syntax



Parameters

-loggerName (logger_name)

Required. The name of the Managed File Transfer logger you want to modify.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to modify the logger. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyLogger** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-s service_name

Optional. Indicates that the logger is to run as a Windows service. If you do not specify *service_name*, the service is named `mqmftLoggerLOGGERQMGR`, where *LOGGER* is the logger name and *QMGR* is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Logger *LOGGER*@*QMGR***.

-su user_name

Required when **-s** is specified. Specifies the name of the account under which the Windows service should run. To run the agent using a Windows domain user account, specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain, specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the permission to log on as a service. For information about how to grant this permission, see [Guidance for running an MFT agent or logger as a Windows service](#).

-sp password

Optional. Only valid when **-s** is specified. Password for the user account set by the **-su** parameter.

If you do not specify this parameter when you specify the **-s** parameter, you are warned that you must set the password by using the Windows Services tool before the service can start successfully.

-sj options

Optional. Only valid when **-s** is specified. When the logger is started as a Windows service, this parameter defines a list of options in the form of `-D` or `-X` that will be passed to the JVM. The options are separated using the number sign (`#`) or semicolon (`;`) character. If you need to embed any `#` or `;` characters, put them inside single quotation marks (`'`).

For more information about the way in which the **fteModifyLogger** command handles the validation of updates to the JVM options see [Guidance for updating agent or logger JVM options](#).

-sl options

Optional. Only valid when **-s** is specified. Sets the Windows service log level. Valid options are: `error`, `info`, `warn`, `debug`. The default is `info`. This option can be useful if you are having problems with the Windows service. Setting it to `debug` gives more detailed information in the service log file.

-n

Optional. Indicates that the logger is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the logger is configured as a normal Windows process.

-? or -h

Optional. Displays command syntax.

Example

You must stop the logger by using the [fteStopLogger](#) command, before running the **fteModifyLogger** command.

In this example, a logger named `logger1` has previously been created. This command shows how the logger can be changed to run as a Windows service:

```
fteModifyLogger -loggerName logger1 -s -su fteuser -sp ftepassword
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[Guidance for running an MFT agent or logger as a Windows service](#)

Related tasks

[Starting an MFT agent as a Windows service](#)

Related reference

[“fteStartLogger” on page 507](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteStopLogger” on page 510](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

fteObfuscate: encrypt sensitive data

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file.

Purpose

User name and password properties in credentials files can be obfuscated. These properties are transformed to a new related property, with a Cipher suffix. For example:

```
<!--
  MQMFTCredentials properties before
-->
<tns:logger name="logger1" user="user1" password="passw0rd"/>
<tns:file path="$HOME/trust.jks" password="passw0rd"/>

<!--
  MQMFTCredentials properties after
-->
<tns:logger name="logger1" userCipher="e71vKCg2pf" passwordCipher="se71vKCg"/>
<tns:file path="$HOME/trust.jks" passwordCipher="e71vKCg2pf"/>

<!--
  ProtocolBridgeCredentials Properties before
-->
<tns:user name="Fred" serverUserId="fred" serverPassword="passw0rd"/>

<!--
  ProtocolBridgeCredentials properties after
-->
<tns:user name="Fred" serverUserIdCipher="e51vVCg2pf" serverPasswordCipher="se51vBCg"/>

<!--
  ConnectDirectCredentials properties before
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
  cdUserId="cdUser" cdPassword="cdPassword" pnodeUserId="pnodeUser"
  pnodePassword="pnodePassword">
  <tns:snode name="snode1" pattern="wildcard" userId="snodeUser" password="snodePassword"/>
</tns:user>

<!--
  ConnectDirectCredentials properties after
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
  cdUserIdCipher="e71vKCg2pf" cdPasswordCipher="se71vKCg"
  pnodeUserIdCipher="2f1vgCg6df" pnodePasswordCipher="e71vKCg2pf">
  <tns:snode name="snode1" pattern="wildcard" userIdCipher="e51vVCg2pf" passwordCipher="se51vBCg"/>
</tns:user>
```

Syntax

fteObfuscate

```
► fteObfuscate — -credentialsFile — credentials_file_name ◄
```

Parameter

-credentialsFile

Required. Name of the credentials file whose contents will be obfuscated.

-? or -h

Optional. Displays command syntax.

Example

In this example, the `MQMFTCredentials.xml` contents are obfuscated.

```
fteObfuscate -credentialsFile /home/fteuser/MQMFTCredentials.xml
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“MFT and IBM MQ connection authentication” on page 39](#)

Connection authentication allows a queue manager to be configured to authenticate applications by using a provided user ID and password. If the associated queue manager has security enabled, and requires credential details (user ID and password), the connection authentication feature must be enabled before a successful connection to a queue manager can be made. Connection authentication can be run in compatibility mode or MQCSP authentication mode.

[“MFT credentials file format” on page 785](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

ftePingAgent: check whether an MFT agent is active

The **ftePingAgent** command pings a Managed File Transfer agent to determine whether the agent is reachable and, if so, whether it is able to respond to a simple query.

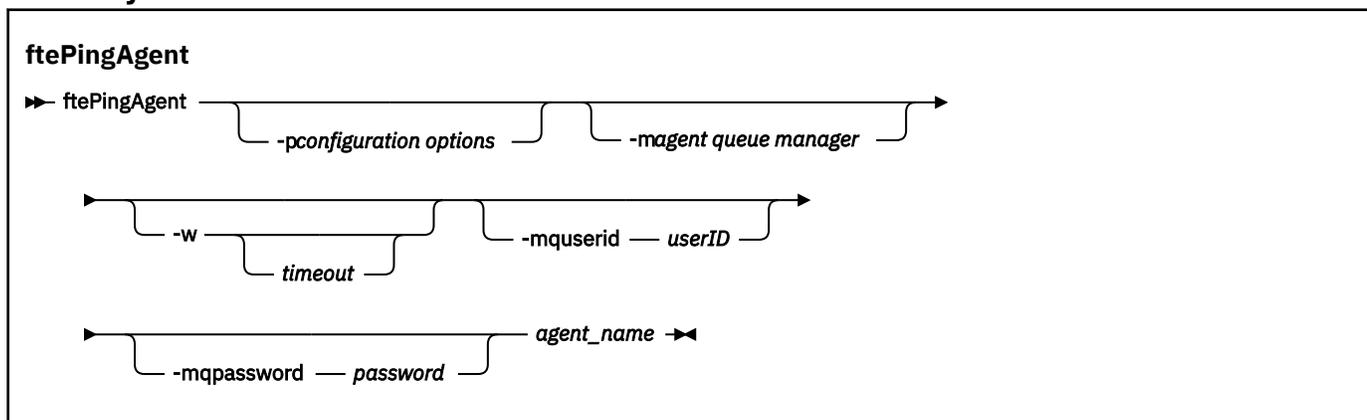
Purpose

Use the **ftePingAgent** command to check whether a Managed File Transfer agent is reachable and, if so, whether it is able to respond to a simple query along the lines of `are you there?`. An example output of this command is as follows:

```
C:\> ftePingAgent AGENT86
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGPR0127W: No credentials file has been specified to connect to IBM MQ. Therefore, the
assumption is that IBM MQ authentication has been disabled.
BFGCL0212I: Issuing ping request to agent AGENT86
BFGCL0213I: agent AGENT86 responded to ping in 0.094 seconds.
```

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

Syntax



Parameters

-p (configuration options)

Optional. This parameter determines the set of configuration options that is used to issue the request to ping an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used. See [Configuration options](#) for more information.

-m (queue manager)

Optional. The name of the queue manager that the agent you want to ping is connected to. If you do not specify the `-m` parameter the queue manager used is determined from the set of configuration options in use.

-w (timeout)

Optional. Specifies that the command should wait for up to *timeout* seconds for the agent to respond. If you do not specify a timeout, or specify a timeout value of `-1`, then the command waits indefinitely until the agent responds. If you do not specify this option then the default is to wait up to five seconds for the agent to respond.

If *timeout* has been specified, **ftePingAgent** command messages will time out after double the value of *timeout* rather than going to the designated dead letter queue. The command messages will not time out if the command has been set to wait indefinitely.

-mquserid (user ID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

(agent name)

Required. The name of the Managed File Transfer agent that you want to ping.

-? or -h

Optional. Displays command syntax.

Example

In this example, the command pings the agent AGENT1, which is connected to QM_MERCURY. The command waits for up to 40 seconds for AGENT1 to respond before returning.

```
ftePingAgent -m QM_MERCURY -w 40 AGENT1
```

Return codes

0

Command completed successfully. The agent is active and able to process transfers.

1

Command ended unsuccessfully. The command was not able to send a message to the agent.

2

Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time.

Related reference

[“fteListAgents: list the MFT agents for a coordination queue manager” on page 452](#)

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“fteShowAgentDetails” on page 495](#)

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

[“What to do if you think that your file transfer is stuck” on page 299](#)

On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

fteRAS: collect MFT troubleshooting information

The **fteRAS** command collects troubleshooting information (MustGather data) for Managed File Transfer. The information that **fteRAS** collects is specific to the Managed File Transfer installation on the system where the program is being run.

Purpose

Use the **fteRAS** command to run the Reliability, Availability, and Serviceability information (RAS) gathering tool if you need to collect troubleshooting information to use to help find a solution when a Managed File Transfer agent, database logger or other command is reporting a problem or failing to work properly.

When you run the **fteRAS** command, the output directory in which the resulting archive (.zip) file is placed can be either the default location, or a directory of your choosing.

Syntax

fteRAS

```
► fteRAS -p configuration_options -l PDSLibraryName ◀
```

Security

```
► -mquserid userid -mqpassword password ◀
```



Parameters

-p *configuration_options*

Optional. Determines the set of configuration options that is used to gather the troubleshooting information, for example, the list of agents. Use the name of a set of configuration options as the value for the **-p** parameter. By convention, this name is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

z/OS -l

Optional. z/OS only. Specifies the name of a PDS library that contains JCL scripts that invoke MQMFT commands for a particular agent or logger. This option is always set when the command is run from a command PDS library's BFGRAS JCL script, such that all members of the PDS library are captured in the output directory.

-mquserid *user id*

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword *password*

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid** but do not also specify **-mqpassword**, you are prompted to supply the associated password. The password is not displayed on the screen.

OutputDirectory

Optional. A directory to use when you are gathering the RAS data, and where the output file, for example, `fteRAS.zip` is stored after the data is gathered successfully. If the directory does not exist, it is created. The default location is the mqft logs directory.

-? or -h

Optional. Displays command syntax.

Examples

Linux **UNIX** On UNIX and Linux, to store the output file `fteRAS.zip` in the `/var/mqm/errors` directory, run **fteRAS** as shown in the following example:

```
fteRAS /var/mqm/errors
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in /var/mqm/errors/fteRAS.zip
```

Windows On Windows, to store the output file `fteRAS.zip` in the default errors directory for a new installation of IBM MQ, run **fteRAS** as shown in the following example:

```
fteRAS "C:\ProgramData\IBM\MQ\errors"
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in  
C:\ProgramData\IBM\MQ\errors\fteRAS.zip
```

Note: For IBM MQ 8.0 or later, if this is not a new installation of that version of the product, the location of the errors directory might be different on your system. For more information, see [Program and data directory locations on Windows](#).

IBM i On IBM i, to copy the output file to `/QIBM/UserData/mqm/errors`, run the **fteRAS** command from the Qshell as shown in the following example:

```
/QIBM/ProdData/mqm/bin/fteRAS /QIBM/UserData/mqm/errors
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in /QIBM/UserData/mqm/errors/
fteRAS.zip
```

Related reference

“Troubleshooting MFT” on page 259

Use the following reference information to help you to diagnose errors in Managed File Transfer:

fteSetAgentLogLevel: set MFT protocol bridge agent log level

Use the **fteSetAgentLogLevel** command to enable or disable diagnostic information logging for file transfers between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers.

Purpose

Use this command to enable or disable logging for collection of diagnostic information about the transfer of files between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers. Managed File Transfer protocol bridge agent logging is more lightweight, separate from, but can be run in parallel with trace.

Commands that are sent by the protocol bridge agent to FTP/SFTP/FTPS file servers, and the responses that are received, are written to a log file for simple diagnosing of problems that might occur during file transfers.

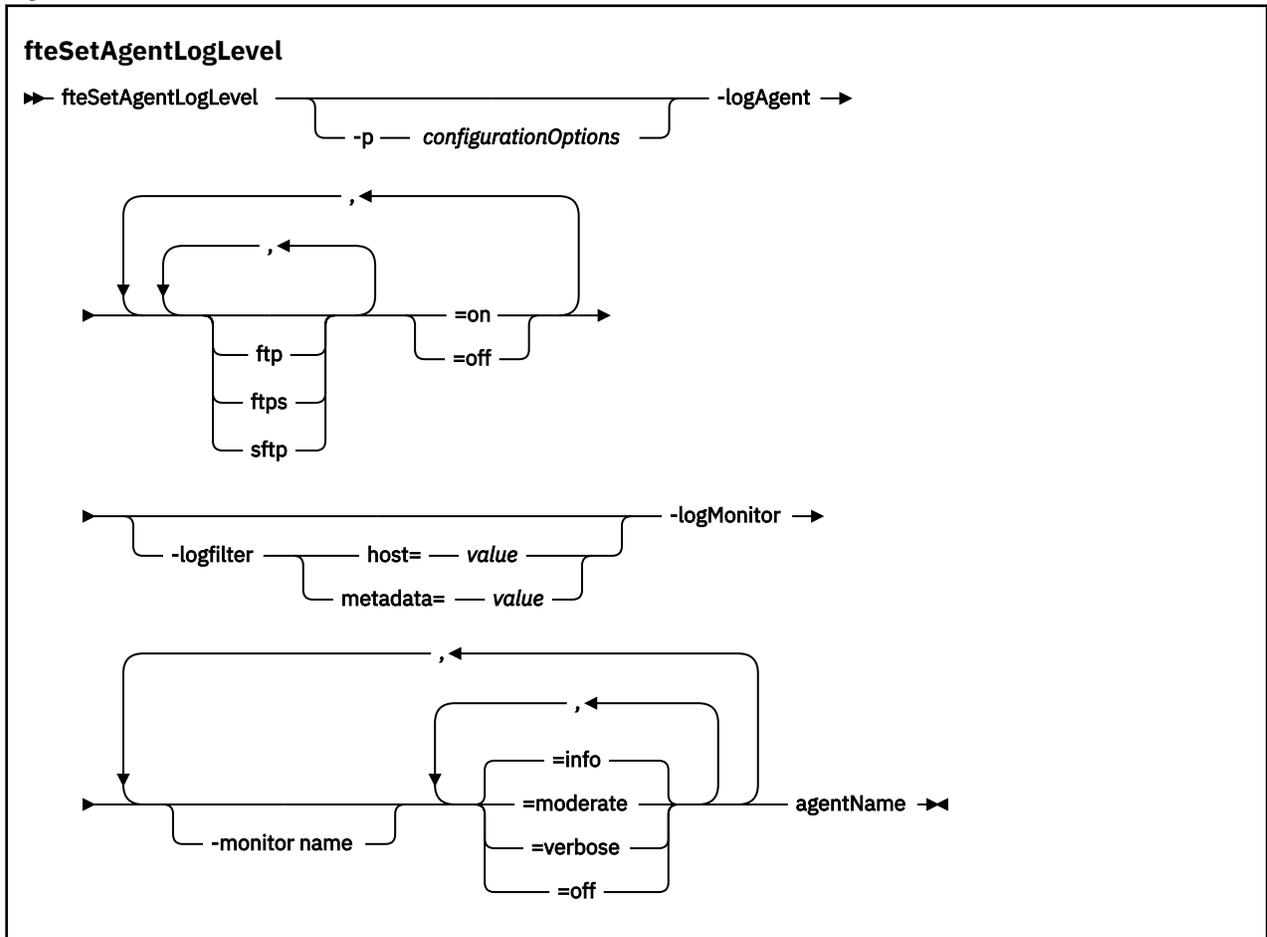
You can enable, disable, and set the level of logging that you require, in two ways:

- Use the **fteSetAgentLogLevel** command to enable or disable logging while the protocol bridge agent is running. You do not need to restart the protocol bridge agent for the logging to begin.
- Use the `agent.properties` file to enable or disable logging at agent start. If you use this method to enable or disable logging, and the agent is running, you must restart the agent for the logging to begin. For more information, see [The agent.properties file](#).

An interaction between a protocol bridge agent and a file server is treated as an event and each event is written to an `agentevent?.log` file. The '?' in the log name represents a numeric value. The file that contains 0 (zero) in its name, contains the newest set of captured messages. The log file is stored in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/` folder.

You can set further logging properties, for example, to limit the log file size and the number of log files kept, in the [The agent.properties file](#).

Syntax



Parameters

-logAgent component=operation

Required.

V 9.0.3

Important: You must select only one of **logAgent** and **logMonitor**. If you attempt to select both options you receive an error message.

Name of the protocol bridge agent for which the logging is enabled or disabled. Protocol bridge agent logging can be enabled or disabled for three components: *ftp*, *ftps*, and *sftp* servers. Specify one of the three possible server types and add an operation value to turn the logging *off* or *on*, for the protocol bridge agent they are connecting to.

V 9.0.3

If a component starts with a plus sign (+), the list of components following the plus sign are added to any existing log component currently being logged.

The valid log level operation options are as follows:

off

Disable all logging for a protocol bridge agent. This option is the default.

```
fteSetAgentLogLevel -logAgent =off PBA1
```

To disable logging for a specified component that the protocol bridge agent is connecting to, use these commands:

```
fteSetAgentLogLevel -logAgent ftp=off PBA1
```

```
fteSetAgentLogLevel -logAgent ftps=off PBA1
```

```
fteSetAgentLogLevel -logAgent sftp=off PBA1
```

on

To enable logging for all three possible file server components that the protocol bridge agent is connecting to, use this command:

```
fteSetAgentLogLevel -logAgent =on PBA1
```

To enable logging for a specified component that a protocol bridge agent is connecting to, use these commands:

```
fteSetAgentLogLevel -logAgent ftp=on PBA1
```

```
fteSetAgentLogLevel -logAgent ftps=on PBA1
```

```
fteSetAgentLogLevel -logAgent sftp=on PBA1
```

For further configuration options, see [“Example 1” on page 482](#) and [“Example 2” on page 482](#).

-agentName

Required. Name of the protocol bridge agent for which the logging is enabled or disabled.

-logFilter filter=value

Optional. Use to limit logging to the specified filter criteria. You must specify a value for either **host** or the user specified file transfer **metadata**.

host

Specify one of three values to filter by: host name, a list of comma separated host names or IP addresses. To log commands for, and responses from FTP server `ftpprod.ibm.com`, use this command:

```
fteSetAgentLogLevel -logAgent ftp=on -logFilter host=ftpprod.ibm.com PBA1
```

To log commands and responses from all SFTP servers with IP addresses starting with `9.182.*`, use this command:

```
fteSetAgentLogLevel -logAgent sftp=on -logFilter host=9.182.* PBA1
```

metadata

Specify any text, as defined by the user during the transfer creation, in a *key=value* format. For example **metadata="BANK=WORLD BANK"**. To enable logging for any FTP server that connects to the protocol bridge agent PBA1 and filter the file transfers to include them only if they contain the **metadata** as key and "BANK=WORLD BANK" as value, use this command:

```
fteSetAgentLogLevel -logAgent ftp=on metadata="BANK=WORLD BANK" PBA1
```

Note: In order to filter by metadata, the value you are filtering by, must be specified under the **-md** parameter as a part of a file transfer. For more information, see [“fteCreateTransfer: start a new file transfer” on page 417](#).

V 9.0.3**-logMonitor *monitor name=log level***

Required.

Important: You must select only one of **logAgent** and **logMonitor**. If you attempt to select both options you receive an error message.

A comma separated list of resource monitors to enable or disable logging, where:

monitor name

Optional. Resource monitor name, or list of names, to enable or disable logging. If you do not specify a name, logging is enabled for all resource monitors of the agent.

log level

Required. One of:

info

Turn on info level logging. This is the default value.

To enable info level logging for monitor MON1 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=info AGENT1
```

moderate

Turn on moderate level logging.

To enable moderate level logging for monitors MON1 and MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1, MON2=moderate AGENT1
```

verbose

To turn on verbose level logging.

Enable verbose level logging for all monitors of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor =verbose AGENT1
```

off

Turn off logging.

To turn off logging for monitors MON1 and MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1, MON2=off AGENT1
```

To turn off logging for monitor MON1 and enable info level logging for monitor MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=off, MON2=info AGENT1
```

To turn off logging for all monitors of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor =off AGENT1
```

V 9.0.3**-p *configurationOptions***

Optional. Determines the set of configuration options that is used to set the agent log level. Use the name of a set of configuration options as the value for the **-p** parameter.

By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-? or -h

Optional. Displays the command syntax.

V 9.0.3

If the same resource monitor name is repeated in a command, then the last occurrence of a component is considered as valid. For example:

```
fteSetAgentLogLevel -logMonitor MON1=info, MON2=off, MON1=off AGENT1
```

turns off logging for resource monitor MON1.

V 9.0.3



Attention: If you have specified non-existent resource monitor names in the command, no error is displayed on the console.

Example 1

In this example, multiple components are specified in one command, by using a command delimited group. Logging is enabled for FTP and SFTP servers but disabled for FTPS servers that the PBA1 protocol bridge agent is connecting to.

```
fteSetAgentLogLevel -logAgent ftp=on,ftps=off,sftp=on PBA1
```

You can also separate the components with a comma to achieve the same result, for example;

```
fteSetAgentLogLevel -logAgent ftp,sftp=on,ftps=off PBA1
```

Example 2

In this example, the same component is repeated in a command. The last instance of a *component=operation* pair is considered as valid. This example disables logging for FTP servers that the PBA1 protocol bridge agent is connecting to.

```
fteSetAgentLogLevel -logAgent ftp=on,ftp=off PBA1
```

The previous example has the same effect as this example:

```
fteSetAgentLogLevel -logAgent ftp=off PBA1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[“The protocol bridge” on page 204](#)

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

fteSetAgentTraceLevel

Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

Purpose

Use this command to switch agent trace on and off or to change the level of agent trace that is set. When you use the **fteSetAgentTraceLevel** command, you do not have to shut down and restart an agent to modify the trace level. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%/trace%PID%.txt`, where `%PID%` is the process ID for the agent instance.



Attention:

Multi When using IBM WebSphere MQ 7.5 or later on Multiplatforms, only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

V 9.0.2 **z/OS** For z/OS, for Continuous Delivery, from IBM MQ 9.0.2, the **fteSetAgentTraceLevel** command can be run by either:

- The same userid that the agent process is running as.
- Members of the group specified by the agent property **adminGroup**.

V 9.0.0.1 This also applies to the Long Term Support release from IBM MQ 9.0.0 Fix Pack 1.

For more information, see the **adminGroup** property in [“The MFT agent.properties file”](#) on page 526.

In IBM WebSphere MQ 7.5, and later, the **fteSetAgentTraceLevel** command also writes a trace for the agent process controller. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/pctrace%PID%/pctrace%PID%.txt`, where `%PID%` is the process ID for the agent instance.

You can also use the command to cause the agent process to generate a Javacore. The agent generates a Javacore file in the following directory: `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name`.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.



Attention:

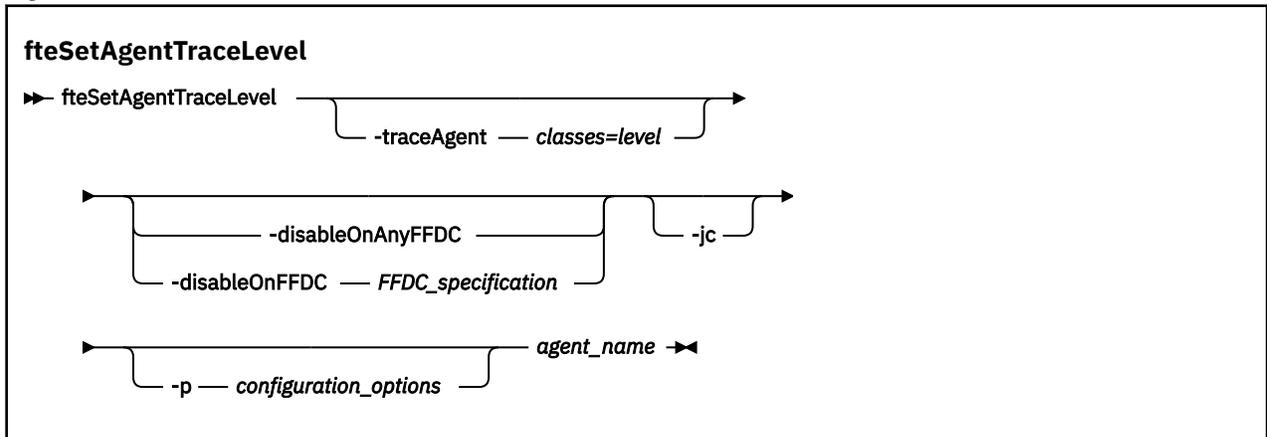
1. You must run this command on the system where the agent is running.
2. The traces and logging do not persist across an agent restart.

If the agent terminates and is restarted by the Process Controller process, the dynamic traces and logs are not in effect until the `agent.properties` file has been updated to include the required trace and log properties.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `agent.properties` file. These properties are described in [Advanced agent properties](#).

Specify the optional `-p` parameter for this command only if you want to use a set of configuration options different from your default set. See [“The MFT agent.properties file”](#) on page 526 for more information.

Syntax



Parameters

-traceAgent classes=level

Required. Level to set the agent trace and which classes to apply the trace to.

V 9.0.5 From IBM MQ 9.0.5, you can specify a colon-separated list of class specifications. This option enables you to set tracing of different classes, and at different levels. For example:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent=all:com.ibm.wmqfte.cmdhandler=moderate AGENT1
```

You can still specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes. Use the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

You can substitute *classes* with a Managed File Transfer package name to trace a specific package only. However, because this option captures just a subset of the agent's behavior, you are generally not recommended to use package filtering.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off

Switches the agent trace off but continues to write information to the log files. This is the default option.

flow

Captures data for trace points associated with processing flow in the agent.

moderate

Captures a moderate amount of diagnostic information in the trace.

verbose

Captures a verbose amount of diagnostic information in the trace.

all

Sets agent trace to run on all agent classes.

To start full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =all AGENT_NAME
```

To stop full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =off AGENT_NAME
```

-disableOnAnyFFDC

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC FFDC_specification

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The format of the values can be either:

class_name

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc

Optional. Requests that the agent generates a Javacore file. The IBM service team may request that you run the command with this parameter to assist with problem diagnosis. This parameter cannot be used with any other parameter except **-p**.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to set the agent trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name

Required. The name of the Managed File Transfer Agent that you want to set the trace level for.

-? or -h

Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.agent.Agent` and `com.ibm.wmqfte.cmdhandler` for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler=moderate AGENT1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to off. All classes starting with com.ibm.outer are traced at verbose level except classes starting with com.ibm.outer.inner:

```
fteSetAgentTraceLevel -traceAgent com.ibm.outer=verbose AGENT1
fteSetAgentTraceLevel -traceAgent +com.ibm.outer.inner=off AGENT1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

fteSetLoggerTraceLevel

Use the **fteSetLoggerTraceLevel** command to modify the current trace level for a Managed File Transfer logger dynamically.

Purpose

Use this command to switch logger trace on and off or change the level of logger trace that is set. When you use the **fteSetLoggerTraceLevel** command, you do not have to shut down and restart a logger to modify the trace level. The trace files that are produced are located in *MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/trace%PID%/trace%PID%.txt*, where *%PID%* is the process ID for the logger instance.

In IBM WebSphere MQ 7.5 and later, the **fteSetLoggerTraceLevel** command also writes a trace for the logger process controller. The trace files that are produced are located in *MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs/pctrace%PID%/pctrace%PID%.txt*, where *%PID%* is the process ID for the logger instance.

The command can also be used to cause the logger process to generate a Javacore. The logger generates a Javacore file in the following directory: *MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name*.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.



Attention:

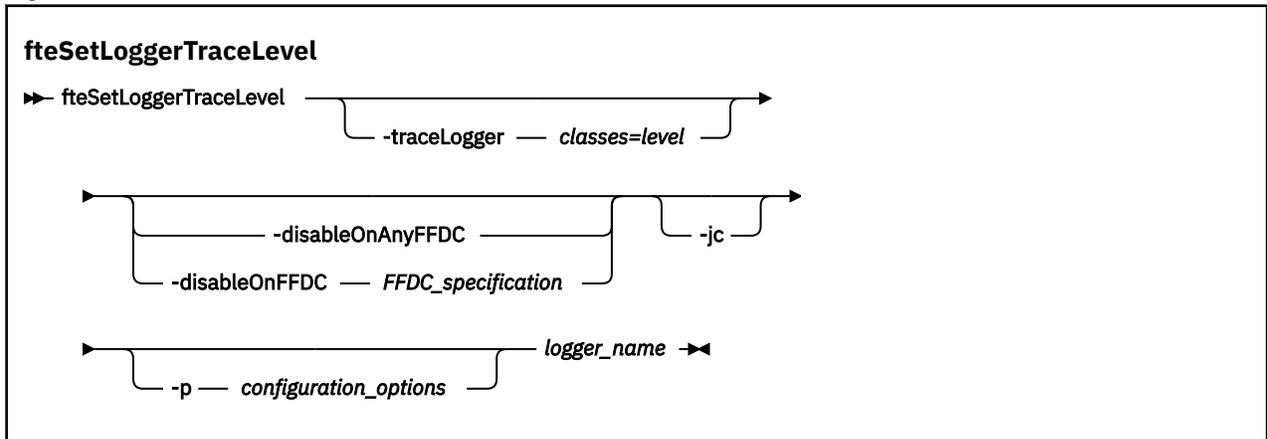
1. You must run this command on the system where the agent is running.
2. The traces and logging do not persist across an agent restart.

If the agent terminates and is restarted by the Process Controller process, the dynamic traces and logs are not in effect until the `agent.properties` file has been updated to include the required trace and log properties.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `logger.properties` file. These properties are described in [Logger properties](#).

Specify the optional `-p` parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [“MFT logger configuration properties” on page 111](#).

Syntax



Parameters

-traceLogger *classes=level*

Required. Level to set the logger trace and which classes to apply the trace to.

V 9.0.5 From IBM MQ 9.0.5, you can specify a colon-separated list of class specifications. This option enables you to set tracing of different classes, and at different levels. For example:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte.logger=all:com.ibm.wmqfte.cmdhandler=moderate
LOGGER1
```

You can still specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes. Use the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all logger classes.

If (*classes*) start with a plus sign (+), the list of trace classes that follow the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off

Switches off the logger trace but continues to write information to the log files. This is the default option.

flow

Captures data for trace points associated with processing flow in the logger.

moderate

Captures a moderate amount of diagnostic information in the trace.

verbose

Captures a verbose amount of diagnostic information in the trace.

all

Sets logger trace to run on all logger classes.

-disableOnAnyFFDC

Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*

Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The value can be one of the following formats:

class_name

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc

Optional. Requests that the logger generates a Javacore file. The IBM service team might request that you run the command with this parameter to assist with problem diagnosis. You cannot use the **-jc** parameter with any other parameter.

-p *configuration_options*

Optional. This parameter determines the set of configuration options that is used to set the logger trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

logger_name

Required. The name of the Managed File Transfer Logger that you want to set the trace level for.

-? or -h

Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for `LOGGER1`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte=all LOGGER1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.logger.logger` and `com.ibm.wmqfte.cmdhandler` for `LOGGER1`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte.logger.logger,com.ibm.wmqfte.cmdhandler=moderate  
LOGGER1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to `off`. All classes that start with `com.ibm.outer` are traced at verbose level except classes that start with `com.ibm.outer.inner`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.outer=verbose LOGGER1  
fteSetLoggerTraceLevel -traceLogger +com.ibm.outer.inner=off LOGGER1
```

Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

V 9.0.3 z/OS **fteSetProductId: set z/OS SCRT recording product id**

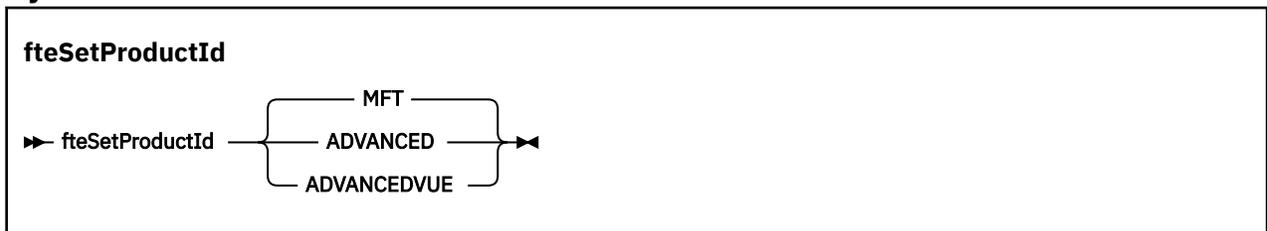
The **fteSetProductId** is used to set the product type against which Managed File Transfer usage is to be recorded for the installation. This command is valid only on z/OS.

Purpose

This command can be run at any time, after at least one coordination queue manager has been defined, or the [installation.properties](#) file, for the installation, has been created.

See [Reporting product information](#) for more information on product usage recording.

Syntax



Parameters

The product type for usage recording:

Specify one of:

MFT

Usage is recorded as a stand-alone Managed File Transfer product, with product ID 5655-MF9.

ADVANCED

Usage is recorded as part of an IBM MQ Advanced for z/OS product, with product ID 5655-AV9.

ADVANCEDVUE

Usage is recorded as part of an IBM MQ Advanced for z/OS, Value Unit Edition product, with product ID 5655-AV1.

Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully, or if the product type has not been set to the requested value.

Related tasks

[“Configuring the coordination queue manager for MFT” on page 92](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the

necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

fteSetupCommands: create the MFT command.properties file

The **fteSetupCommands** command creates the Managed File Transfer `command.properties` file. This properties file specifies the details of the queue manager that connects to the IBM MQ network when you issue commands.

Important: **ULW** On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message `BFGCL0502E: You are not authorized to perform the requested operation.` and the command will not run.

z/OS On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the migrate command:

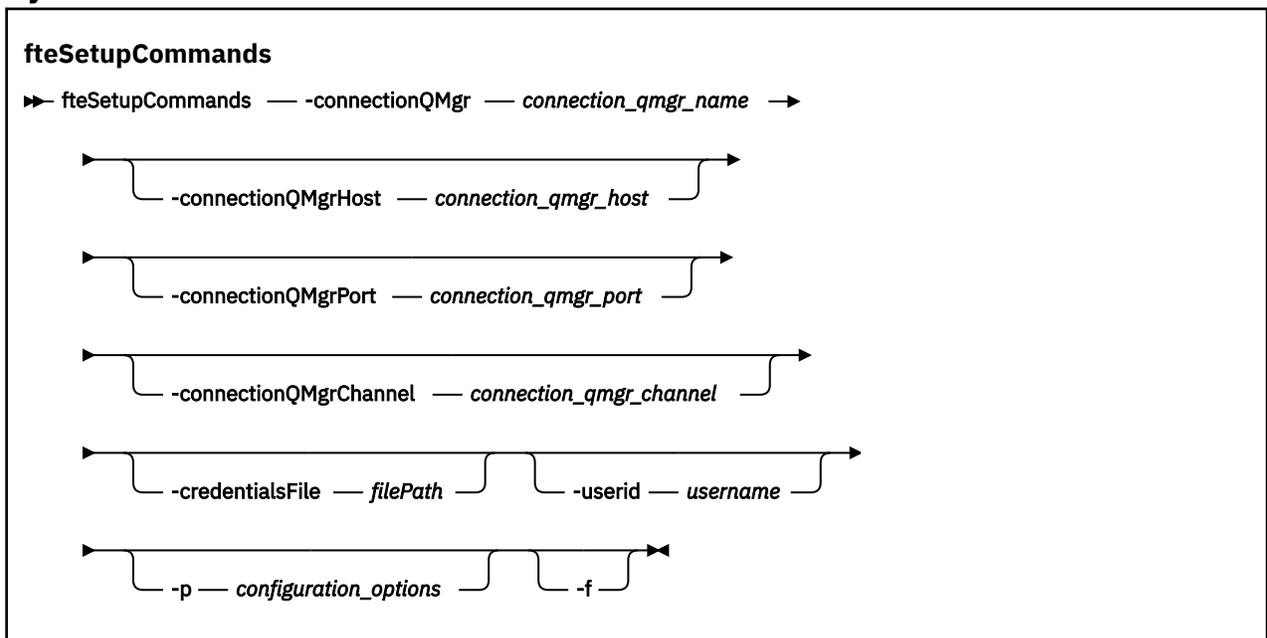
- Be a member of the `mqm` group (if the `mqm` group is defined on the system).
- **V9.0.0.1** Be a member of the group named in the `BFG_GROUP_NAME` environment variable (if one is named).
- **V9.0.0.1** Have no value set in the `BFG_GROUP_NAME` environment variable when the command is run.

Purpose

Use the **fteSetupCommands** command to create a `command.properties` file in the coordination queue manager configuration directory. The command uses the `MQ_DATA_PATH` environment variable and the `installation.properties` file to determine where to locate the `command.properties` file. Ensure that you have already created and configured a coordination queue manager before you issue the **fteSetupCommands** command.

For more information about properties files, see [“The MFT command.properties file” on page 522.](#)

Syntax



Parameters

-connectionQMgr (*connection_qmgr_name*)

Required. The name of the queue manager used to connect to the IBM MQ network to issue commands.

-connectionQMgrHost (*connection_qmgr_host*)

Optional. The host name or IP address of the connection queue manager.

If you do not specify the **-connectionQMgrHost** parameter, a bindings mode connection is assumed. Therefore, this parameter is required if you are using a client mode connection.

If you specify a value for the **-connectionQMgrHost** parameter but do not specify values for the **-connectionQMgrPort** and **-connectionQMgrChannel** properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

-connectionQMgrPort (*connection_qmgr_port*)

Optional. The port number used to connect to the connection queue manager in client mode. If you specify the **-connectionQMgrPort** parameter, you must also specify the **-connectionQMgrHost** parameter.

-connectionQMgrChannel (*connection_qmgr_channel*)

Optional. The channel name used to connect to the connection queue manager. If you specify the **-connectionQMgrChannel** parameter, you must also specify the **-connectionQMgrHost** parameter.

-p (*configuration_options*)

Optional. This parameter determines the set of configuration options that is used to set up a command queue manager. Use the name of a non-default coordination queue manager as the input for this parameter. The **fteSetupCommands** command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-credentialsFile (*filePath*)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-userid (*username*)

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

-f

Optional. Forces an overwrite of the existing `command.properties` file with the details specified in this command.

-? or -h

Optional. Displays command syntax.

Example

```
fteSetupCommands -connectionQMgr QM_NEPTUNE -connectionQMgrHost 9.146.157.241  
-connectionQMgrPort 1414 -connectionQMgrChannel SYSTEM.DEF.SVRCONN
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“The MFT command.properties file” on page 522](#)

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

[“fteSetupCoordination” on page 492](#)

The **`fteSetupCoordination`** command creates properties files and the coordination queue manager directory for Managed File Transfer.

fteSetupCoordination

The **`fteSetupCoordination`** command creates properties files and the coordination queue manager directory for Managed File Transfer.

Important:  On IBM MQ for UNIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message `BFGCL0502E: You are not authorized to perform the requested operation.` and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the `migrate` command:

- Be a member of the `mqm` group (if the `mqm` group is defined on the system).
-  Be a member of the group named in the `BFG_GROUP_NAME` environment variable (if one is named).
-  Have no value set in the `BFG_GROUP_NAME` environment variable when the command is run.

Purpose

Use the **`fteSetupCoordination`** command to create the following Managed File Transfer objects:

- Coordination queue manager directory
- Data directory `mqft` (if this does not exist)
- `installation.properties` file
- `coordination.properties` file

This command also provides you with the following MQSC commands that you must run against your coordination queue manager to configure Managed File Transfer. The MQSC commands create a topic, a topic string, the `SYSTEM.FTE` queue, and the default database logger queues. These commands also update a namelist and set the `PSMODE` attribute of the coordination queue manager to `ENABLED`.

 If the coordination queue manager is on z/OS, before you run these MQSC commands, you must ensure that the following required objects already exist:

- `SYSTEM.BROKER.DEFAULT.STREAM` queue
- `SYSTEM.QPUBSUB.QUEUE.NAMELIST` namelist
- `SYSTEM.BROKER.DEFAULT.STREAM` and `SYSTEM.BROKER.ADMIN.STREAM` streams

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
ALTER TOPIC('SYSTEM.FTE') NPMSGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)
```

```

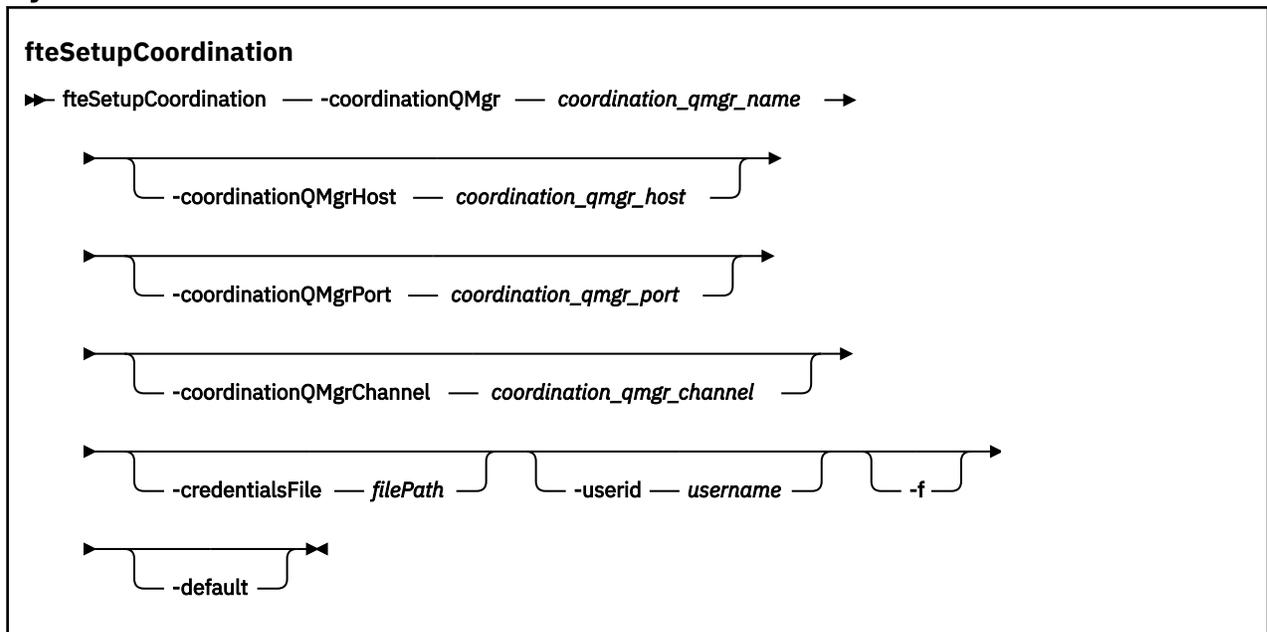
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for WMQFTE Pub/Sub interface')
* Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST
* Value prior to alteration:
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +
  NAMES(SYSTEM.BROKER.DEFAULT.STREAM+
,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
* Altering PSMODE. Value prior to alteration:
DISPLAY QMGR PSMODE
ALTER QMGR PSMODE(ENABLED)

```

For more information about properties files, see [Configuration options](#).

z/OS If you are using z/OS, you can issue the **fteSetupCoordination** command and other commands from JCL with scripts generated from the Managed File Transfer command template PDSE library data set. For more information, see [“Creating an MFT Agent or Logger command data set”](#) on page 62.

Syntax



Parameters

-coordinationQMgr (*coordination_qmgr_name*)

Required. The name of the coordination queue manager. This queue manager must be an IBM WebSphere MQ 7.0 or later queue manager.

-coordinationQMgrHost (*coordination_qmgr_host*)

Optional. The host name or IP address of the coordination queue manager.

If you do not specify the **-coordinationQMgrHost** parameter, a bindings mode connection is assumed.

If you specify a value for the **-coordinationQMgrHost** parameter but do not specify values for the **-coordinationQMgrPort** and **-coordinationQMgrChannel** parameters, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

-coordinationQMgrPort (*coordination_qmgr_port*)

Optional. The port number used for client connections to the coordination queue manager.

If you specify the **-coordinationQMgrPort** parameter, you must also specify the **-coordinationQMgrHost** parameter.

-coordinationQMGrChannel (*coordination_qmgr_channel*)

Optional. The channel name used to connect to the coordination queue manager. If you specify the **-coordinationQMGrChannel** parameter, you must also specify the **-coordinationQMGrHost** parameter.

-credentialsFile (*filePath*)

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

-userid (*username*)

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

-f

Optional. Forces an overwrite of the existing coordination queue manager configuration with the details specified in this command.

-default

Optional. Updates the default configuration options to the options associated with the coordination queue manager specified in this command.

-? or -h

Optional. Displays command syntax.

Example

In this example, the required objects are set up for a coordination queue manager called QM_SATURN, which is connected to in client mode:

```
fteSetupCoordination -coordinationQMGr QM_SATURN  
-coordinationQMGrHost myhost.ibm.com -coordinationQMGrPort 1415  
-coordinationQMGrChannel SYSTEM.DEF.SVRCONN
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

[“Configuring MQMFTCredentials.xml on z/OS” on page 42](#)

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT agents, and commands that connect to a queue manager, to supply user ID and password credentials.

Related tasks

[“Configuring the coordination queue manager for MFT” on page 92](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the

necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Related information

[“SSL properties for the coordination.properties file” on page 582](#)

fteShowAgentDetails

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

Purpose

You can run the **fteShowAgentDetails** command from any system that can connect to the coordination queue manager. This command uses the `coordination.properties` file to connect to the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [“MFT configuration options on Multiplatforms” on page 56](#).

The agent status information that is produced by this command is generated from the status messages that the agent publishes to the `SYSTEM.FTE` topic. These messages are described in [“MFT agent status message format” on page 709](#). The status information that is produced by the **fteShowAgentDetails** command gives the agent status at the time when the last status message was published. The frequency of these status messages depends on the value of the **agentStatusPublishRateLimit** property. For more information, see [“The MFT agent.properties file” on page 526](#).

For IBM WebSphere MQ 7.5.0 Fix Pack 1 or later, specify the optional **-d** parameter for this command if you want to see diagnostic information about a local agent. This information includes current transfers, scheduled transfers, monitors, and agent queue depths. You can use this information to determine the health and status of a local agent.

 For z/OS, from IBM MQ 9.0.2 and IBM MQ 9.0.0 Fix Pack 1, the **-d** parameter can only be specified if the **fteShowAgentDetails** command is run by:

- The same userid that the agent process is running as.
- Members of the group that is specified by the agent property **adminGroup**.

For more information, see the **adminGroup** property in [“The MFT agent.properties file” on page 526](#).

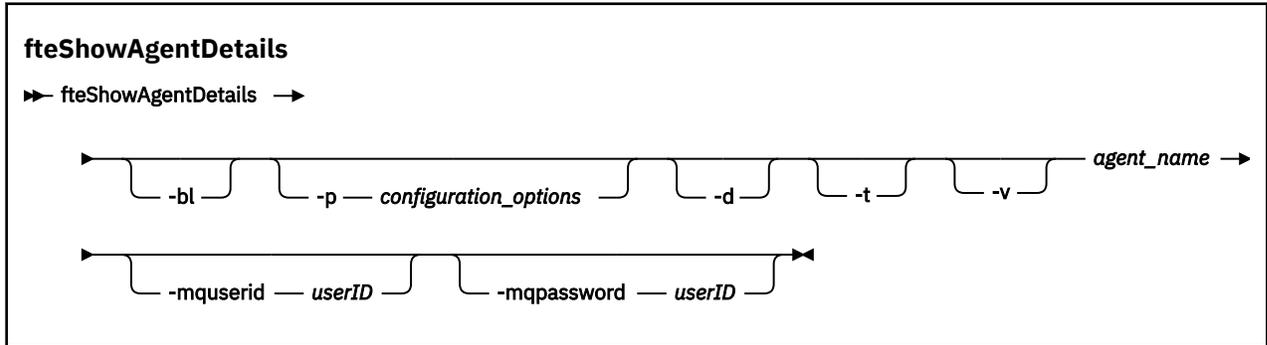
For IBM WebSphere MQ 7.5, or later, the status of the agent process controller and queue manager is available if you run the command on the same system as the agent. You can use this information to help with problem determination. Also, when you run the command on the same system as the agent, more detailed agent status information is available for the case where the agent ended unexpectedly.

For a list of the possible agent status values and their meanings, see [“MFT agent status values” on page 599](#).

For a list of the possible status values for the agent process controller and their meanings, see [“MFT agent process controller status values” on page 602](#).

For a list of agent trace values and FFDC specifications and their meanings, see [“fteSetAgentTraceLevel” on page 262](#)

Syntax



Parameter

-bl

Optional. Additionally outputs the product build level for the agent.

-p (*configuration_options*)

Optional. This parameter determines the set of configuration options that is used to issue the request to display the details of an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-d

Optional. Specifies that diagnostic information is collected for *agent_name*.

The diagnostic information is output to the console, and written to a file called `diagnostics.<yyyyMMdd>.<HHmmss>.<ssss>.<number>.properties` in the directory `MQ_DATA_PATH\mqft\logs\coordination_qmgr_name\agents\agent_name\logs`. A maximum of five historical files containing diagnostic information about an agent will be created. If five historical files have been created for an agent when the **fteShowAgentDetails** command is run with the **-d** parameter specified, the oldest historical file will be deleted and replaced with a new file containing the latest diagnostic information about the agent.

You can use this parameter only when the agent is running, and on the local system.

-t

Optional. Specifies terse mode. At IBM MQ 9.0.2, the output includes the **Status Age** information by default. If you do not want to see this information, you can issue the command with the **-t** parameter. For more information, see [“What to do if an agent is shown as being in an UNKNOWN state” on page 290](#).

-v

Optional. Specifies verbose mode, which generates additional output for the agent. These include host name, product version, product build level, trace level, and First Failure Data Capture (FFDC) specification, and a list of transfer states for each of the current source and destination transfers.

The current transfer information is obtained from the agent status publication, which is described in [“MFT agent status message format” on page 709](#). Therefore this transfer information is only accurate to within the value of the `agentStatusPublishRateLimit` property. For more details about this property, see [“The MFT agent.properties file” on page 526](#).

agent_name

Required. The name of the Managed File Transfer Agent that you want to display.

-mquserid (*userID*)

Optional. Specifies the user ID to authenticate with the coordination queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the coordination queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

-? or -h

Optional. Displays command syntax.

Example

In the following example, running bindings agent, issuing the **fteShowAgentDetails** command locally to the agent:

```
fteShowAgentDetails -v AGENT1
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
```

Agent Information:

```
Name: AGENT1
Type: Standard
Description:
Operating System: Windows Server 2003
Time Zone: Greenwich Mean Time
Product Version: 7.5
Build Level: f000-20120312-0957
Trace Level: com.ibm.wmqfte.Agent=all
              com.ibm.wmqfte.common=all
              com.ibm.wmqfte.common:Any
              com.ibm.wmqfte.Agent:1

Trace FFDC:
```

Agent Controller Information:

```
Controller type: MQMFT Process Controller
Status: STARTED
Status Details: The agent process controller has
                started the agent process.

Agent Restarts within Interval: 0
Total Agent Restart Count: 0
```

Agent Availability Information:

```
Status: READY
Status Details: The agent is running and is publishing
                its status at regular intervals. The
                last update was received within the
                expected time period. The agent is
                ready to process transfers, but none
                are currently in progress.
```

Queue Manager Information:

```
Name: QM1
Transport: Bindings
Last Status Reported: AVAILABLE (Last Error MQRC: 0)
Status Details: The queue manager is available.
```

```
Maximum Number of Running Source Transfers: 25
Maximum Number of Queued Source Transfers: 1000
Source Transfer States:
  No current transfers
```

```
Maximum Number of Running Destination Transfers: 25
Destination Transfer States:
```

TransferId	State
414d51204d49414f572020202020202020822c5b4a648c0b20	progress
414d51204d49414f572020202020202020822c5b4a346c0b20	progress

In the following example, QMGR1 is the non-default coordination queue manager used as input for the configuration options, and diagnostic information is requested with the **-d** parameter. The **fteShowAgentDetails** command is issued on an IBM WebSphere MQ 7.5.0 Fix Pack 1 system with a local agent:

```
fteShowAgentDetails -p QMGR1 -d AGENT1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
Name: AGENT1
Type: Standard
```

```

Description:
Operating System:      Linux
Time Zone:            Greenwich Mean Time

Agent Controller Information:
Controller type:      MQMFT Process Controller
Status:              STARTED
Status Details:      The agent process controller has started
                    the agent process.

Agent Restarts within Interval: 0
Total Agent Restart Count: 0

Agent Availability Information:
Status:              ACTIVE
Status Details:      The agent is running and is publishing
                    its status at regular intervals. The last
                    update was received within the expected
                    time period. The agent is currently
                    processing one or more transfers.

Queue Manager Information:
Name:                QMGR1
Transport:           Client
Host:                host1.hursley.ibm.com
Port:                2021
Channel:             SYSTEM.DEF.SVRCONN
Last Status Reported: UNKNOWN
Status Details:      Information about the queue manager is
                    not available because the agent has a
                    client connection to the queue manager.

Agent Diagnostic Information:

Command Handler Diagnostics:
Last Command Queue Read Time: 2012-07-30T15:23:10.705Z
Pending Command Queue Size: 0
Last Internal Command Type: Resync Request (from sender) -
414d5120514d43414e4445202020202079e20f5064230010
Last Internal Command Time: 2012-07-30T14:17:10.506Z
Last External Command Type: New Monitor Request
Last External Command Time: 2012-07-30T14:10:57.751Z
Diagnostic Properties File name: C:\Program Files (x86)\IBM\WebSphere
MQ\mqft\logs\MUNGEE\agents\MUNGEE\logs\di
agnostics.20121031.083420.0477.1.properties

Command Handler Worker Thread 0 Diagnostics:
Status:              Waiting

Command Handler Worker Thread 1 Diagnostics:
Status:              Waiting

Command Handler Worker Thread 2 Diagnostics:
Status:              Waiting

Command Handler Worker Thread 3 Diagnostics:
Status:              Waiting

Command Handler Worker Thread 4 Diagnostics:
Status:              Waiting

File Transfer Diagnostics:
Source Transfers:    1
Destination Transfers: 2

File Transfer 0 Diagnostics:
Transfer Id:         414d5120514d43414e4445202020202079e20f5064230010
Role:                SOURCE
State:               ReSynchronisingTransfer
Status:              INACTIVE
Start Time:         Not started
Retry Count:         0
CheckPoint Index:    0
CheckPoint Position: 0

File Transfer 1 Diagnostics:
Transfer Id:         414d5120514d43414e44452020202020c8fbd54f144f0d20
Role:                DESTINATION
State:               RunningTransfer
CheckPoint Index:    0
CheckPoint Position: 0
Write Index:         0

```

```

Write Position:                0

File Transfer 2 Diagnostics:
Transfer Id:                   414d5120514d43414e4445202020202079e20f5086020010
Role:                         DESTINATION
State:                         RunningTransfer
CheckPoint Index:             9
CheckPoint Position:         0
Write Index:                   3
Write Position:               140923

Monitor 0 Diagnostics:
Name:                          MONITOR1
Status:                        STARTED
Resource Type:                 directory
Resource:                      /tmp/monitor
Poll Interval:                 1 minutes
Batch Size:                    2
Condition:                     Match
Pattern:                       * (wildcard)
Executing:                     false
Last Execute Start Time:      2012-04-04T16:19:01.852Z
Last Execute End Time:        2012-04-04T16:19:01.852Z
Last Execute Match Count:     0

Schedule 1 Diagnostics:
Id:                             1
Next Trigger Time:             2012-07-17T16:00+0100
Occurrences So Far:            14
Repeat Interval:               hours
Repeat Frequency:              5
Source Agent:                  AGCANDE
Destination Agent:             AGCANDE
Source File:                   /tmp/source/a.txt, ...
Destination File:              /tmp/dest/a.txt, ...

```

In the following example, stopped bindings agent, issuing the **fteShowAgentDetails** command remotely from the agent:

```

fteShowAgentDetails AGENT2
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                        AGENT2
  Type:                        Standard
  Description:
  Operating System:            Linux
  Time Zone:                   Greenwich Mean Time

Agent Controller Information:
  Controller type:             MQMFT Process Controller
  Status:                      UNKNOWN
  Status Details:             Information about the agent controller
                              is not available, either because the
                              agent is not running or the agent is
                              running on a different system.

  Agent Restarts within Interval: 0
  Total Agent Restart Count:    0

Agent Availability Information:
  Status:                      STOPPED
  Status Details:              The agent has been stopped. It was shut
                              down in a controlled manner.

Queue Manager Information:
  Name:                        QM2
  Transport:                   Bindings
  Last Status Reported:        UNKNOWN
  Status Details:             Information about the queue manager is
                              not available, either because the agent
                              is not running or the agent is running
                              on a different system.

```

In the following example, bindings agent is waiting to restart with the agent queue manager stopped. The agent has already been restarted once before Total Agent Restart Count: 1, possibly due to a previous agent queue manager restart:

Note: The Last Error MQRC against the Last Status Reported for the queue manager information; this information will remain even when the queue manager becomes available.

```
fteShowAgentDetails AGENT1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT1
  Type:                Standard
  Description:
  Operating System:    Windows Server 2003
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              WAITING
  Status Details:      The agent process controller is waiting
                       for the queue manager to become
                       available before starting the agent.

  Agent Restarts within Interval: 0
  Total Agent Restart Count:      1

Agent Availability Information:
  Status:              STOPPED
  Status Details:      The agent has been stopped. It was shut
                       down in a controlled manner.

Queue Manager Information:
  Name:                QM1
  Transport:           Bindings
  Last Status Reported: UNAVAILABLE (Last Error MQRC: 2059)
  Status Details:      The queue manager is unavailable. It
                       might be that the queue manager has not
                       been started or an incorrect queue
                       manager name has been configured. Look
                       up the MQ reason code reported against
                       the status to understand the problem.
```

In the following example, the client mode agent has just ended unexpectedly and the agent process controller tries to recover the situation by restarting it after a delay, specified by the `maxRestartDelay` agent property value. The default `maxRestartDelay` agent property value is `-1`, and this causes the agent process controller to terminate; hence in this example the `maxRestartDelay` property value must have been set to a value greater than `0`. The `Current Agent Restart Count: 4` implies that there have been 4 restarts within the `maxRestartInterval` agent property time period. If the `maxRestartCount` agent property is `4` then after 4 restarts within the `maxRestartInterval`, the agent process controller will wait for `maxRestartDelay` seconds before restarting the agent, which is the case here. The `Total Agent restart Count: 8` suggests that this has occurred before. This example is not typical and you would only expect to see the agent ending unexpectedly if the agent runs out of memory, or a custom user exit has caused some sort of runtime error. Full details as to why the agent ended unexpectedly are in the agent's `output0.log` file:

```
fteShowAgentDetails AGENT3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT3
  Type:                Standard
  Description:
  Operating System:    Windows Server 2003
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              RECOVERING
  Status Details:      The agent process unexpectedly stopped
                       and the process controller will attempt
                       to restart it.

  Current Agent Restart Count: 4
  Total Agent Restart Count:  8

Agent Availability Information:
  Status:              ENDED UNEXPECTEDLY
  Status Details:      The agent has ended unexpectedly due to
                       an unrecoverable problem. The agent
                       will be automatically restarted.

Queue Manager Information:
```

```

Name: QM3
Transport: Client
Host: host3.hursley.ibm.com
Port: 3031
Channel: SYSTEM.DEF.SVRCONN

```

In the following example, the results for a Connect:Direct bridge agent are displayed:

```

fteShowAgentDetails AG_CD1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name: AG_CD1
  Type: Connect:Direct bridge
  Description:
  Connect:Direct Node Name: CDNODE
  Connect:Direct Node Host: localhost:1363
  Operating System: Windows Server 2003
  Time Zone: Greenwich Mean Time

Agent Controller Information:
  Controller type: MQMFT Process Controller
  Status: UNKNOWN
  Status Details: Information about the agent controller
                 is not available, either because the
                 agent is not running or the agent is
                 running on a different system.

  Agent Restarts within Interval: 0
  Total Agent Restart Count: 0

Agent Availability Information:
  Status: STOPPED
  Status Details: The agent has been stopped. It was shut
                 down in a controlled manner.

Queue Manager Information:
  Name: QM_JUPITER
  Transport: Bindings
  Last Status Reported: UNKNOWN
  Status Details: Information about the queue manager is
                 not available, either because the agent
                 is not running or the agent is running
                 on a different system.

```

 In the following example, an agent running on z/OS is registered with the Automatic Restart Manager (ARM):

```

fteShowAgentDetails AGENTZ
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name: AGENTZ
  Type: Standard
  Description:
  Operating System: z/OS
  Time Zone: Greenwich Mean Time

Agent Controller Information:
  Controller Type: z/OS Automatic Restart Manager (ARM)
  Agent registered with ARM: Yes (ELEMENTYPE: SYSBFGAG, ELEMENT: AGENTZ)
  Agent Restarted: No

Agent Availability Information:
  Status: READY
  Status Details: The agent is running and is publishing
                 its status at regular intervals. The last
                 update was received within the expected
                 time period. The agent is ready to
                 process transfers, but none are currently
                 in progress.

Queue Manager Information:
  Name: ZQM
  Transport: Bindings
  Last Status Reported: AVAILABLE
  Status Details: The queue manager is available.

```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“fteListAgents: list the MFT agents for a coordination queue manager” on page 452](#)

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“MFT agent status values” on page 599](#)

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

[“MFT agent process controller status values” on page 602](#)

The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

fteShowLoggerDetails

Use the **fteShowLoggerDetails** command to display the details of a particular Managed File Transfer logger.

Purpose

You must run the **fteShowLoggerDetails** command on the same system as the logger. It displays the status of the logger process controller and the logger queue manager, which you can use to help with problem determination. The **fteShowLoggerDetails** command lists the following details for a particular Managed File Transfer logger:

- Logger controller status.
- Logger restarts within interval
- Total logger restart count
- Logger availability status
- Logger queue manager name
- Logger queue manager transport type
- Logger queue manager last status reported (applies to binding transport mode only)

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [“MFT configuration options on Multiplatforms” on page 56](#) for more information.

For a list of the possible logger status values and their meanings, see [“MFT logger status values” on page 602](#).

For a list of the possible status values for the logger process controller and their meanings, see [“MFT logger process controller status values” on page 603](#).

V 9.0.4 From IBM MQ 9.0.4, the output of the command displays connection information that the logger is using to connect to the queue manager.

V 9.0.4 If the logger is connected in clients mode, the output for:

Last Status Reported

Is shown as UNKNOWN

Status details

Is shown as Information about the queue manager is not available because the logger has a client connection to queue manager.

Syntax

fteShowLoggerDetails

```
► fteShowLoggerDetails ———— configuration_options ———— logger_name ◄◄
```

Parameter

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to issue the request to display the details of an logger. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

logger_name

Required. The name of the Managed File Transfer logger that you want to display.

-? or -h

Optional. Displays command syntax.

Example

In this example, a started logger, issuing the **fteShowLoggerDetails** command locally to the logger:

```
fteShowLoggerDetails LOGGER1
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Status:                               STARTED
  Status Details:                        The logger process controller has
                                          started the logger process.
  Logger Restarts within Interval:       0
  Total Logger Restart Count:           0

Queue Manager Information:
  Name:                                  QM_gbthink
  Transport:                             Bindings
  Last Status Reported:                  AVAILABLE
  Status Details:                        The queue manager is available.
```

In this example, a logger waiting due to an unavailable queue manager, issuing the **fteShowLoggerDetails** command locally to the logger:

```
fteShowLoggerDetails LOGGER2
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Status:                               WAITING
  Status Details:                        The logger process controller is
                                          waiting for the queue manager to
                                          become available before starting the
                                          logger.
  Logger Restarts within Interval:       0
  Total Logger Restart Count:           0

Logger Availability Information:
  Status:                               STOPPED
  Status Details:                        The logger has been stopped. It was
                                          shut down in a controlled manner.
```

```

Queue Manager Information:
  Name:                QM_gbthink
  Transport:           Bindings
  Last Status Reported: UNAVAILABLE (Last Error MQRC: 2059)
  Status Details:      The queue manager is unavailable. It
                       might be that the queue manager has
                       not been started or an incorrect
                       queue manager name has been
                       configured. Look up the MQ reason code
                       reported against the status to
                       understand the problem.

```

In this example on z/OS, a running logger (not registered with ARM):

```
fteShowLoggerDetails loggerv8
```

```

5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Controller Type:      z/OS Automatic Restart Manager (ARM)
  Registered with ARM: No
  Restarted:           n/a

Queue Manager Information:
  Name:                FT8E
  Transport:           Bindings
  Last Status Reported: AVAILABLE
  Status Details:      The queue manager is available.

```

In this example on z/OS, a logger that is not running, or running on a different system:

```
fteShowLoggerDetails loggerv8
```

```

5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Controller Type:      UNKNOWN

Queue Manager Information:
  Name:                FT8E
  Transport:           Bindings
  Last Status Reported: UNKNOWN
  Status Details:      Information about the queue manager is
                       not available, either because the
                       logger is not running, or the logger
                       is running on a different system.

```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related reference

[“MFT logger status values” on page 602](#)

The **fteShowLoggerDetails** commands produce logger status information. There are several possible values for this status.

[“MFT logger process controller status values” on page 603](#)

The **fteShowLoggerDetails** command produces logger process controller status information. There are several possible values for this status.

fteStartAgent

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

Purpose

Use the **fteStartAgent** command to start a Managed File Transfer agent. You must start an agent before you can use it to perform file transfers. The **fteStartAgent** command starts an agent on the system where you issue the command: you cannot start an agent on a remote system.

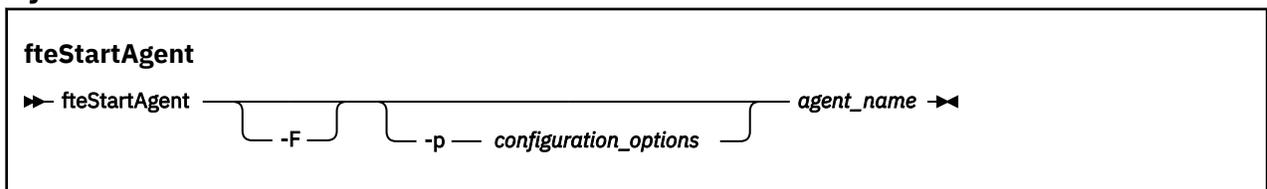
For IBM WebSphere MQ 7.5 or later, the agent process controller manages starting the agent. However, the agent process controller may wait for a period of time, for example where there have been a high rate of agent failures, before attempting to start the agent again. As an IBM MQ administrator you can use **fteStartAgent** command to override this wait and initiate a start of the agent. If the agent process controller was waiting for the queue manager to become available this command will also initiate the agent process controller attempting to reconnect to the queue manager.

Windows If you have configured the agent to run as a Windows service by using the [fteCreateAgent](#) or [fteModifyAgent](#) command, running the **fteStartAgent** command starts the Windows service.

This command returns an error if the agent does not start or is already started. The agent communicates with its queue manager based on the values defined in the `agent.properties` file.

Specify the optional **-p** parameter for this command only if you want to use a different set of configuration options than your default set. See [“The MFT agent.properties file” on page 526](#) for more information.

Syntax



Parameter

-F

Optional. This parameter runs the agent daemon as a foreground process. The default is for the agent daemon to run in the background.

If you are running on Windows, and you have configured the agent to run as a Windows service by using the **fteCreateAgent** or **fteModifyAgent** commands, the **-F** parameter overrides this configuration.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to issue the request to start an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name

Required. The name of the Managed File Transfer agent that you want to start.

-? or -h

Optional. Displays command syntax.

Example

In this example, AGENT2 is started and runs in the foreground.

```
fteStartAgent -F AGENT2
```

In the following example (for UNIX and Linux systems), AGENT2 is started with a non-default coordination queue manager, QM_SATURN:

```
./fteStartAgent -p QM_SATURN AGENT2
```

You can also run the command by specifying the path to **fteStartAgent** as follows:

```
path/fteStartAgent agentname
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Responses

In some circumstances, you might see error messages after running the **fteStartAgent** command:

- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with Managed File Transfer:

```
BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:  
The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

If the LD_LIBRARY_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java (as is currently the case for most platforms), this error occurs.

To resolve this issue, set the Managed File Transfer agent property javaLibraryPath to reference the correct location for the library. For example, for mqjbnd on AIX, set to: /usr/mqm/java/lib. For mqjbnd on Linux, set to: /opt/mqm/java/lib

Related tasks

[“Starting an MFT agent as a Windows service” on page 147](#)

You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

[“Listing MFT agents” on page 203](#)

You can list the Managed File Transfer agents registered with a particular queue manager using the command line or the IBM MQ Explorer.

[“Stopping an MFT agent” on page 203](#)

You can stop a Managed File Transfer agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

Related reference

[“Starting an MFT agent on z/OS” on page 150](#)

On z/OS, in addition to running the **fteStartAgent** command from a UNIX System Services session, you can start an agent as a started task from JCL without the need for an interactive session.

fteStartLogger

The **fteStartLogger** command starts a Managed File Transfer logging application.

Purpose

Use the **fteStartLogger** command to start a logger. The logger can be either a file or database application that runs on the same system as the coordination queue manager. For more information, see the topic “Configuring an MFT logger” on page 100. For IBM WebSphere MQ 7.5, or later, the logger process controller manages starting the logger. However, the logger process controller may wait for a period of time, for example where there have been a high rate of logger failures, before attempting to start the logger again. As an IBM MQ administrator you can use the **fteStartLogger** command to override this wait and initiate a start of the logger. If the logger process controller was waiting for the queue manager to become available this command will also initiate the logger process controller attempting to reconnect to the queue manager.

If you have configured a logger to run as a Windows service by using the **fteModifyLogger** command, running the **fteStartLogger** command starts the Windows service.

This command returns an error if the logger does not start or is already started. The logger communicates with its queue manager based on the values defined in the `logger.properties` file.

Specify the **-p** parameter for this command only if you want to use a set of configuration options different from the default. For more information on logger properties, see “MFT logger configuration properties” on page 111

Syntax

fteStartLogger

```
► fteStartLogger -p configuration_options -F logger_name ◄
```

Parameters

logger_name

Required. The name of the Managed File Transfer logger you want to start.

-p configuration_options

Optional. This parameter determines the set of configuration options that is used to issue the request to start a logger. Use the name of a non-default coordination queue manager as the input for this parameter. **fteStartLogger** then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-F

Optional. Runs the logger as a foreground process (rather than as the default background process). If you have configured the logger to run as a Windows service by using the **fteModifyLogger** command, the **-F** parameter overrides this configuration.

-? or -h

Optional. Displays command syntax.

Example

In this example, a logger has previously been created named `logger1`. This command shows how the logger can be started as a foreground process:

```
fteStartLogger -F logger1
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[“Configuring an MFT logger” on page 100](#)

Related reference

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the `mqm` group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStopLogger” on page 510](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

[“MFT logger error handling and rejection” on page 315](#)

The Managed File Transfer logger identifies two types of error: per-message errors and general errors.

fteStopAgent

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

Purpose

When you stop an agent by using the **fteStopAgent** command, you can either allow the agent to complete its current file transfer before stopping, or stop the agent immediately even if the agent is currently transferring a file. When the agent has stopped, you cannot use that agent to transfer files until you restart the agent.

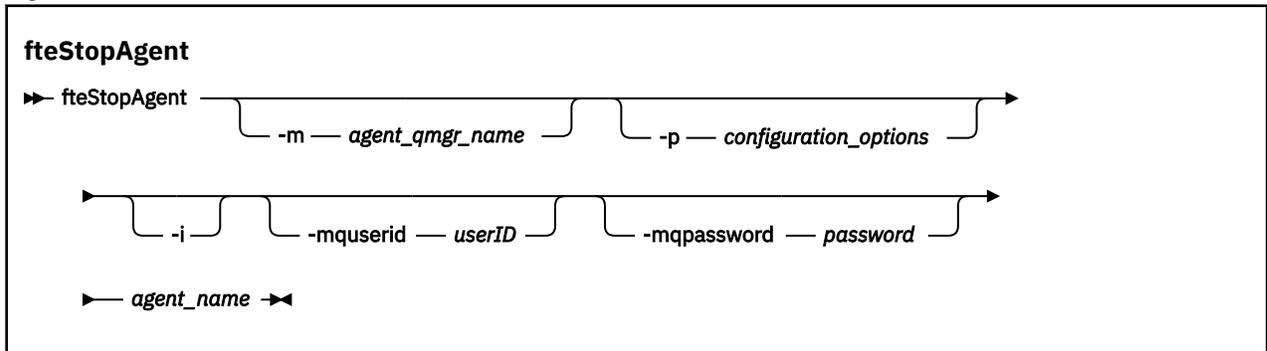
If the agent you want to stop is connected to the IBM MQ network, you can run the **fteStopAgent** command from any system that can connect to the IBM MQ network and route to the agent queue manager. Specifically for the command to run, you must have installed and configured a Managed File Transfer component (either Service or Agent) on this system to communicate with the IBM MQ network. If no connectivity details are available, a bindings mode connection is made to the default queue manager on the local system. If `command.properties` does not exist then an error is generated.

If the agent you want to stop is not connected to the IBM MQ network, for example if the IBM MQ network is not currently available, you can only run the **fteStopAgent** command from the system that the agent is running on. In order to stop an agent that is not connected to the IBM MQ network you must run the **fteStopAgent** command from the same user the agent is running as. Alternatively, if the agent is running on a Windows system you can run the command as an administrator.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [“The MFT agent.properties file” on page 526](#) for more information.

If your agent is running as a Windows service, running the **fteStopAgent** command stops the Windows service. For more information, see [“Starting an MFT agent as a Windows service” on page 147](#).

Syntax



Parameters

-m (*agent_qmgr_name*)

Optional. The name of the queue manager that the agent that you want to stop is connected to.

If the agent is on a remote system, or if the agent is on the local system but you are not the user that started it, you must use the **-m** parameter, and have the appropriate authorities. For more information about authorities, see [“Restricting group authorities for MFT-specific resources”](#) on page 337.

-p (*configuration_options*)

Optional. This parameter determines the set of configuration options that is used to issue the request to stop an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-i

Optional. Immediately stops the agent. The agent does not complete any transfers that are currently in progress.

If you do not specify the **-i** parameter, the agent completes any transfers currently in progress but the agent does not start any new transfers.

-mquserid (*userID*)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (*password*)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

agent_name

Required. The name of the Managed File Transfer agent that you want to stop.

-? or -h

Optional. Displays command syntax.

Example

In this example the agent AGENT2 on queue manager QM_JUPITER is stopped. The **-m** parameter is used because this queue manager that AGENT2 is connected to differs from the queue manager specified by the set of configuration options.

```
fteStopAgent -m QM_JUPITER AGENT2
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Related tasks

[“Stopping an MFT agent” on page 203](#)

You can stop a Managed File Transfer agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

Related reference

[“fteStartAgent” on page 505](#)

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

[“Stopping an MFT agent on z/OS” on page 203](#)

If you are running a Managed File Transfer Agent on z/OS as a started task from JCL, the agent accepts the z/OS operator commands **MODIFY** and **STOP**, in addition to the **fteStopAgent** command.

fteStopLogger

The **fteStopLogger** command stops a Managed File Transfer logger.

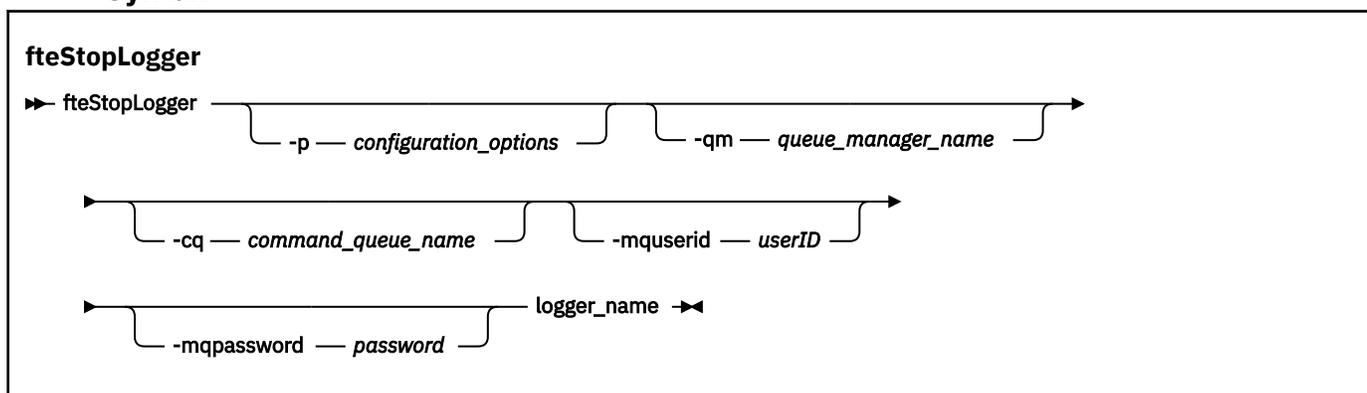
Purpose

Use the **fteStopLogger** command to stop a logger. The logger can be either a file logger, which records a history of managed file transfer activity to a file, or a database logger which records the history to a database.

Additional notes about stopping a stand-alone logger process

If your logger is running as a Windows service, running the **fteStopLogger** command stops the Windows service.

Syntax



Parameters

-p (configuration_options)

Optional. Determines the set of configuration options that is used to stop the logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-qm (queue_manager_name)

Optional. By default, the logger's command queue is assumed to be on the coordination queue manager specified by the **-p** parameter (or its default). If you want to send logger commands to a command queue located elsewhere, use the **-qm** parameter to specify an alternative destination. In all cases, this command connects to the command queue manager indicated by the **-p** parameter, regardless of the message's ultimate destination.

-cq (command_queue_name)

Optional. Specifies the command queue to send the stop message to. In most cases, loggers use the default queue name meaning this parameter is not necessary.

-mquserid (userID)

Optional. Specifies the user ID to authenticate with the command queue manager.

-mqpassword (password)

Optional. Specifies the password to authenticate with the command queue manager. You must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you will be prompted to supply the associated password. The password will not be displayed.

logger_name

Required. The name of the Managed File Transfer logger you want to stop.

-? or -h

Optional. Displays command syntax.

Example

In this example, a logger has previously been created named `logger1` and is currently running. This command shows how the logger can be stopped:

```
fteStopLogger logger1
```

Return codes**0**

Command completed successfully.

1

Command ended unsuccessfully.

Related concepts

[“Configuring an MFT logger” on page 100](#)

Related reference

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the `mqm` group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStartLogger” on page 507](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

MFT configuration reference**The use of environment variables in MFT properties**

From IBM WebSphere MQ 7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used

when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

The following properties accept file or directory locations and can therefore contain environment variables:

- agentSslKeyStore
- agentSslKeyStoreCredentialsFile
- agentSslTrustStore
- agentSslTrustStoreCredentialsFile
- cdNodeKeystoreCredentialsFile
- cdNodeTruststoreCredentialsFile
- cdTmpDir
- cdNodeKeystore
- cdNodeTruststore
- commandPath
- connectionSslKeyStore
- connectionSslKeyStoreCredentialsFile
- connectionSslTrustStore
- connectionSslTrustStoreCredentialsFile
- coordinationSslKeyStore
- coordinationSslKeyStoreCredentialsFile
- coordinationSslTrustStore
- coordinationSslTrustStoreCredentialsFile
- exitClassPath
- exitNativeLibraryPath
- javaCoreTriggerFile
- sandboxRoot
- transferRoot
- wmqfte.database.credentials.file

Example

In this example on a Windows system, a user `fteuser` using an environment variable of `USERPROFILE`:

```
wmqfte.database.credentials.file=%USERPROFILE%\logger\mqmftcredentials.xml
```

Resolves to the following file path:

```
C:\Users\fteuser\logger\mqmftcredentials.xml
```

In this example on a UNIX system, a user `fteuser` using an environment variable of `HOME`:

```
transferRoot=$HOME/fte/
```

Resolves to the following file path:

```
/home/fteuser/fte/
```

The MFT installation.properties file

The `installation.properties` file specifies the name of your default set of configuration options. This entry points Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager.

This file is created by the installer, and can be changed by using the **`fteChangeDefaultConfigurationOptions`** command.

The `installation.properties` file is located in your `MQ_DATA_PATH` directory. For example on Windows, the default file location is `MQ_DATA_PATH\mqft\installations\installation_name` and on UNIX and Linux systems, the default file location is `/var/mqm/mqft/installations/installation_name`.

V 9.0.1 For the Redistributable Managed File Transfer Agent, the data path is set when you run the **`fteCreateEnvironment`** command. If you run the command and specify your chosen location with the **`-d`** parameter, the data path is set for this location. If you do not specify the location with the **`fteCreateEnvironment`** command, a directory `mftdata` is created under the root directory where the Redistributable Managed File Transfer Agent is extracted. The `installation.properties` file for the Redistributable Managed File Transfer Agent is located in the `MQ_DATA_PATH\mqft\installations\MFTZipInstall` directory.

The `installation.properties` file contains the following values:

Table 39. Basic properties

Property name	Description	Default value
commandMessagePriority	<p>Sets the priority of both internal messages and command messages for the fteStopAgent, fteCancelTransfer and ftePingAgent commands.</p> <p>If you submit a large number of transfer requests to transfer many small files in quick succession, for example, the new transfer requests can become queued on the source agent's command queue. The external and internal messages have the default IBM MQ message priority so the internal messages are blocked by the new transfer requests. This can cause the transfer negotiation time to be exceeded and for the transfers to go into recovery.</p> <p>You can also use the <code>commandMessagePriority</code> property to set the priority of internal acknowledgment and acknowledgment-expected messages.</p> <p>To prioritize the internal Managed File Transfer messages above new transfer requests, set this property to a value between 1 (the lowest) and 9 (the highest).</p> <p>The default value of the <code>commandMessagePriority</code> property is 8. This means that, if the IBM MQ attribute <code>DEFPRTY</code> (default priority) on an agent command queue is less than or equal to 7, internal negotiation messages are prioritized ahead of new transfer requests. If the value of the <code>DEFPRTY</code> attribute is set to either 8 or 9, to maintain the effectiveness of the <code>commandMessagePriority</code> property, you must change either <code>DEFPRTY</code> or the <code>commandMessagePriority</code> property.</p>	<p>For IBM MQ 9.0.0.0, the default value is 8.</p> <p>For earlier releases, and before APAR IT06213, the default value is the <code>MQPRI_PRIORITY_AS_Q_DEF</code> constant, which has a value of -1.</p>

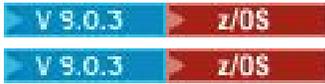
Table 39. Basic properties (continued)

Property name	Description	Default value
defaultProperties	The name of the default set of configuration options. This value is the name of a directory located in the configuration directory, which contains directories and properties files that specify configuration information.	No default
enableFunctionalFixPack	<p>The fix pack function level to enable. By default, any new function included with a fix pack is not enabled. Set this property to a version identifier to enable the new features available with that version.</p> <p>You can specify the version identifier with or without period characters (.). For example, to use the function available with IBM MQ 8.0.0 Fix Pack 2, set this property to 8002 or 8.0.0.2.</p>	No default

Table 39. Basic properties (continued)

Property name	Description	Default value
<p>     messagePublicationFormat </p>	<p>Allows you to specify the message publication format used by MFT agents for their status XML messages. This property can be set to the following values:</p> <p>messagePublicationFormat=mixed</p> <p>Messages are published without an MQMD FORMAT (MQFMT_NONE), except for those messages that are published under the /LOG topic tree, which are published in the MQMD format of MQFMT_STRING.</p> <p>messagePublicationFormat=MQFMT_NONE</p> <p>Messages are published without an MQMD FORMAT.</p> <p>messagePublicationFormat=MQFMT_STRING</p> <p>Messages are published in a string format.</p> <p> There is a coding error in IBM MQ 9.0.2. In that version, when you specify messagePublicationFormat=mixed the effect is the same as specifying messagePublicationFormat=MQFMT_NONE.  The error is fixed in IBM MQ 9.0.3.</p> <p>Before IBM MQ 8.0, MFT agents published XML status messages to the SYSTEM.FTE topic in a string format (MQFMT_STRING). If possible, applications that previously used IBM WebSphere MQ 7.5 must be updated to process messages in the IBM MQ 8.0 or later format. If it is not possible to change an application, set the messagePublicationFormat property to MQFMT_STRING to revert to the IBM WebSphere MQ 7.5 behavior.</p>	<p>messagePublicationFormat=mixed</p>
<p>   z/OS-specific: </p>		

Table 39. Basic properties (continued)

Property name	Description	Default value
 productId	Product type against which MFT usage is to be recorded: <ul style="list-style-type: none"> • Standalone Managed File Transfer product. (MFT is the productId). • Part of an IBM MQ Advanced product. (ADVANCED is the productId). • Part of an IBM MQ Advanced for z/OS, Value Unit Edition product. (ADVANCEDVUE is the productId). See Reporting product information for more information on product usage recording.  This property is ignored on Multiplatforms .	MFT

The following text is an example of the contents of a `installation.properties` file.

```
defaultProperties=ERIS
```

ERIS is the name of a directory that is located in the same directory as the `installation.properties` file. The directory ERIS contains directories and properties files that describe a set of configuration options.

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference

[“fteChangeDefaultConfigurationOptions” on page 365](#)

Use the **`fteChangeDefaultConfigurationOptions`** command to change the default configuration options that you want Managed File Transfer to use. The value of the configuration options defines the group of properties files that Managed File Transfer uses.

The MFT `coordination.properties` file

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

The `coordination.properties` file is created by the installer or by the **`fteSetupCoordination`** command. You can use the **`fteSetupCoordination`** command with the **`-f`** flag to change the basic coordination queue manager properties in this file. To change or add advanced coordination queue manager properties you must edit the file in a text editor.

The `coordination.properties` file is located in your `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory.

The `coordination.properties` file contains the following values:

<i>Table 40. Coordination queue manager properties</i>		
Property name	Description	Default value
coordinationQMgr	The name of the coordination queue manager.	No default
coordinationQMgrHost	The host name or IP address of the coordination queue manager.	No default
coordinationQMgrPort	The port number used for client connections to the coordination queue manager.	1414
coordinationQMgrChannel	The SVRCONN channel name used to connect to the coordination queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the coordinationQMgrHost property, bindings mode is used by default.

If you specify a value for the coordinationQMgrHost property but do not specify values for the coordinationQMgrPort and coordinationQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

<i>Table 41. Advanced coordination queue manager properties</i>		
Property name	Description	Default value
Agent properties:		

Table 41. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
agentStatusJitterTolerance	<p>The maximum amount of time an agent status message publication can be delayed by before the message is considered as overdue. This value is measured in milliseconds.</p> <p>The age of a status message is based on the time at which it was published at the coordination queue manager. However, the message is emitted by the agent some time before it is received at the coordination queue manager to allow for the time required to travel across the IBM MQ network. If this transit always takes the same amount of time, messages created 60 seconds apart are published 60 seconds apart, regardless of the actual time in transit. However, if the transit time varies between messages, they might be created at 60-second intervals but published at intervals of, for example, 61, 59, 58, and 62 seconds. The maximum deviation from 60, 2 seconds in this example, is the jitter. This property determines the maximum delay due to jitter before the message is treated as overdue.</p>	3000
Code page properties:		
coordinationCcsid	<p>The code page the commands connect to the coordination queue manager with. Also any publications to the coordination queue manager made by the agent are performed with this code page. If you specify a value for coordinationCcsid you must also specify a value for coordinationCcsidName.</p>	1208

Table 41. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
coordinationCcsidName	The Java representation of the coordinationCcsid. If you specify a value for coordinationCcsidName you must also specify a value for coordinationCcsid.	UTF8
Connection properties:		
javaLibraryPath	When connecting to a queue manager in bindings mode Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location defined by IBM MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.	<i>MQ_INSTALLATION_PATH/</i> <i>java/lib</i>
Multi-instance queue manager properties:		
coordinationQMGrStandby	The host name and the port number used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance coordination queue manager defined by the coordinationQMGr property. For example, <i>host_name(port_number)</i>	No default
Queue properties:		
dynamicQueuePrefix	<p>This property defines the IBM MQ prefix to use for generating a temporary queue name.</p> <p>The format of the dynamicQueuePrefix property follows the format of the DynamicQName field of the IBM MQ MQOD structure. For more information, see Creating dynamic queues.</p> <p>You can also define this property in the <code>command.properties</code> file if you want to use a specific IBM MQ prefix for temporary reply queues that are generated by commands that require a response from the agent.</p>	WMQFTE.*

Table 41. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
modelQueueName	<p>This property defines the IBM MQ model queue to use for generating a temporary queue.</p> <p>You can also define this property in the command.properties file if you want to use a specific IBM MQ model queue for temporary reply queues that are generated by commands that require a response from the agent. For more information, see “The MFT command.properties file” on page 522.</p>	SYSTEM.DEFAULT.MODEL.QUEUE
Security properties:		
userIdForClientConnect	<p>The user ID that gets flowed through the client connections to IBM MQ. If <i>java</i> is specified the user name reported by the JVM is flowed as part of the IBM MQ connection request. The value of this property can be None or java.</p>	None
coordinationQMgrAuthenticationCredentialsFile	<p>The path to the file that contains the MQ connection credentials for connection to the coordination queue manager.</p>	<p> For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42.</p> <p> See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.</p> <p> Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication</p>
Subscription properties:		

Table 41. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
coordinationSubscriptionTopic	<p>Use this property to specify a topic other than SYSTEM.FTE to subscribe to in order to obtain publications about the status of the IBM MQ network. All tooling still publishes to the SYSTEM.FTE topic, but you can change your IBM MQ topology to distribute these publications to different topics based on their content. You can then use this function to force the tooling to subscribe to one of these other topics.</p> <p>For IBM WebSphere MQ 7.5 and later fix packs, you require an interim fix for APAR IC96850 for the property to be recognized by IBM MQ Explorer and the fteListMonitors command.</p>	SYSTEM.FTE

The following text is an example of the contents of a `coordination.properties` file.

```
coordinationQMGr=ERIS
coordinationQMGrHost=kuiper.example.com
coordinationQMGrPort=2005
coordinationQMGrChannel=SYSTEM.DEF.SVRCONN
```

ERIS is the name of a IBM MQ queue manager that is located on the system `kuiper.example.com`. The queue manager ERIS is the queue manager that Managed File Transfer sends log information to.

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference

[“fteSetupCoordination” on page 492](#)

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for Managed File Transfer.

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

The MFT command.properties file

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

The `command.properties` file is created by the installer or by the **fteSetupCommands** command. You can use the **fteSetupCommands** command with the **-f** flag to change the basic command queue manager properties in this file. To change or add advanced command queue manager properties you must edit the file in a text editor.

Some Managed File Transfer commands connect to the agent queue manager or coordination queue manager instead of the command queue manager. For information about which commands connect to which queue manager, see [“Which MFT commands and processes connect to which queue manager”](#) on page 349.

The `command.properties` file is located in your `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory.

The `command.properties` file contains the following values:

<i>Table 42. Basic command queue manager properties</i>		
Property name	Description	Default value
connectionQMGr	The name of the queue manager used to connect to the IBM MQ network.	No default
connectionQMGrHost	The host name or IP address of the connection queue manager.	No default
connectionQMGrPort	The port number used to connect to the connection queue manager in client mode.	1414
connectionQMGrChannel	The SVRCONN channel name used to connect to the connection queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the `connectionQMGrHost` property, `bindings` mode is used by default.

If you specify a value for the `connectionQMGrHost` property but do not specify values for the `connectionQMGrPort` and `connectionQMGrChannel` properties, a port number of 1414 and a channel of `SYSTEM.DEF.SVRCONN` are used by default.

<i>Table 43. Advanced command queue manager properties</i>		
Property name	Description	Default value
Code page properties:		
connectionCcsid	The code page the commands connect to the command queue manager with. If you specify a value for <code>connectionCcsid</code> you must also specify a value for <code>connectionCcsidName</code> .	1208
connectionCcsidName	The Java representation of the <code>connectionCcsid</code> . If you specify a value for <code>connectionCcsidName</code> you must also specify a value for <code>connectionCcsid</code> .	UTF8
Multi-instance queue manager properties:		
connectionQMGrStandby	The host name and the port number used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance command queue manager defined by the <code>connectionQMGr</code> property. For example, <code>host_name(port_number)</code>	No default

Table 43. Advanced command queue manager properties (continued)

Property name	Description	Default value
Security properties:		
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM MQ. If <i>java</i> is specified the user name reported by the JVM is flowed as part of the IBM MQ connection request. The value of this property can be None or <i>java</i> .	None
connectionQMGrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the command queue manager.	<p> For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42</p> <p> See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.</p> <p> Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication</p>
Queue properties:		
dynamicQueuePrefix	<p>For commands that require a response from the agent, this property defines the IBM MQ prefix to use for generating the temporary reply queue name.</p> <p>The format of the <code>dynamicQueuePrefix</code> property follows the format of the DynamicQName field of the IBM MQ MQOD structure. For more information, see Creating dynamic queues.</p> <p>You can also define this property in the <code>coordination.properties</code> file if you want to use a specific IBM MQ prefix for temporary queues that are generated by WMQFTE.</p>	WMQFTE.*

Table 43. Advanced command queue manager properties (continued)

Property name	Description	Default value
modelQueueName	<p>For commands that require a response from the agent, this property defines the IBM MQ model queue to use for generating the temporary reply queue.</p> <p>You can also define this property in the coordination.properties file if you want to use a specific IBM MQ model queue for temporary queues that are generated by WMQFTE. For more information, see “The MFT coordination.properties file” on page 517.</p>	SYSTEM.DEFAULT.MODEL.QUEUE
Connection properties:		
javaLibraryPath	<p>When connecting to a queue manager in bindings mode Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location defined by IBM MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.</p>	/opt/mqm/java/lib
 legacyXMLMessageMQMDFormat	<p>Managed File Transfer command XML messages are now sent to a queue with a blank MQMD format field. Previous versions of the product set the MQMD format field to MQSTR (a text message string). Setting this property to true enables the Managed File Transfer command XML messages to be sent to a queue with MQMD format field of MQSTR.</p> <p>If the MQMD format field is set to MQSTR, there is potential for Managed File Transfer command XML messages to be corrupted if there are channels in the MQ network with data conversion enabled.</p>	false
Agent properties		

Table 43. Advanced command queue manager properties (continued)

Property name	Description	Default value
 failCleanAgentWithNoArguments	By default, the value of this property is true, which means that the fteCleanAgent command fails to run if only the agent name parameter is specified. Setting the property to false means that, if only the agent name parameter is set, the behavior of the fteCleanAgent command is equivalent to specifying the -all parameter.	true

The following text is an example of the contents of a `command.properties` file.

```
connectionQMgr=PLUTO
connectionQMgrHost=kuiper.example.com
connectionQMgrPort=1930
connectionQMgrChannel=SYSTEM.DEF.SVRCONN
```

PLUTO is the name of an IBM MQ queue manager that is located on the system `kuiper.example.com`. The queue manager PLUTO is the queue manager that the Managed File Transfer commands connect to.

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference

[“Java system properties for MFT” on page 580](#)

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

[“fteSetupCommands: create the MFT command.properties file” on page 490](#)

The **fteSetupCommands** command creates the Managed File Transfer `command.properties` file. This properties file specifies the details of the queue manager that connects to the IBM MQ network when you issue commands.

[“fteCleanAgent: clean up an MFT Agent” on page 366](#)

Use the **fteCleanAgent** command to clean up the queues that a Managed File Transfer Agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

The MFT agent.properties file

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

The `agent.properties` file is created by the installer or by the **fteCreateAgent**, **fteCreateBridgeAgent** or **fteCreateCDAgent** command. You can use any of these commands with the **-f** flag to change the basic agent queue manager properties and those advanced agent properties

that are associated with the type of agent that you are creating. To change or add advanced agent properties, you must edit the file in a text editor.

The `agent.properties` file for an agent is in your `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory.

z/OS On z/OS, the `agent.properties` file location is `$BFG_CONFIG variable/mqft/config/coordination_qmgr_name/agents/agent_name`.

If you change the `agent.properties` file you must restart the agent to pick up the changes.

For IBM WebSphere MQ 7.5, or later, you can use environment variables in some Managed File Transfer properties that represent file or directory locations. This allows you to use the locations of files or directories when running parts of the product to vary depending on environment changes, such as which user is running the process. For more information, see [“The use of environment variables in MFT properties” on page 511](#).

Basic agent properties

Each `agent.properties` file contains the following basic properties.

Property name	Description	Default value
<code>agentName</code>	The name of the agent. The name of the agent must conform to the IBM MQ object naming conventions. For more information, see “MFT object naming conventions” on page 598 .	No default
<code>agentDesc</code>	The description of the agent - if you choose to create a description.	No default
<code>agentQMgr</code>	The agent queue manager name.	No default
<code>agentQMgrHost</code>	The host name or IP address of the agent queue manager.	No default
<code>agentQMgrPort</code>	The port number that is used for client connections to the agent queue manager.	1414
<code>agentQMgrChannel</code>	The SVRCONN channel name that is used to connect to the agent queue manager.	SYSTEM.DEF.SVRCONN
<code>agentType</code>	The type of agent: <ul style="list-style-type: none"> • Standard non-bridge agent (STANDARD) • Protocol bridge agent (BRIDGE) • Connect:Direct bridge agent (CD_BRIDGE) • Embedded agent as used by IBM Integration Bus (EMBEDDED) • Sterling File Gateway embedded agent (SFG) 	STANDARD

If you do not specify a value for the `agentQMgrHost` property, bindings mode is used by default.

If you specify a value for the `agentQMgrHost` property but do not specify values for the `agentQMgrPort` and `agentQMgrChannel` properties, a port number of 1414 and a channel of `SYSTEM.DEF.SVRCONN` are used by default.

Advanced agent properties

Managed File Transfer also provides more advanced agent properties that help you configure agents. If you want to use any of the following properties, manually edit the `agent.properties` file to add the required advanced properties. When you specify file paths on Windows, ensure the separator character

backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\). Alternatively, you can use a single forward slash (/) character as a separator. For more information about character escaping in Java properties files, see the Oracle documentation [Javadoc for the Properties class](#).

- [Agent size properties](#)
- [Code page properties](#)
- [Command properties](#)
- [Connection properties](#)
- [Connect:Direct bridge properties](#)
- [File to message and message to file agent properties](#)
- [General agent properties](#)
- [Input/output properties](#)
- [Multi-channel support properties](#)
- [Multi-instance properties](#)
- [Process controller properties](#)
- [Protocol bridge properties](#)
- **V 9.0.0** [Protocol bridge agent log properties](#)
- [Queue properties](#)
- [Resource monitoring properties](#)
- [Root directory properties](#)
- [Scheduler property](#)
- [Security properties](#)
- [Timeout properties](#)
- **V 9.0.1** [Transfer recovery timeout properties](#)
- [Trace and logging properties](#)
- [Transfer limit properties](#)
- [User exit routine properties](#)
- [IBM MQ client compression properties](#)
- **z/OS** [z/OS-specific properties](#)
- [Other properties](#)

<i>Table 45. Advanced agent properties</i>		
Property name	Description	Default value
Agent size properties:		
agentCheckpointInterval	<p>The interval in complete frames of data between which a checkpoint is taken for recovery purposes. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value.</p> <p>If there is a problem which causes the transfer to go into recovery, the transfer can recover only to a checkpoint boundary. Hence, the larger this value (with large agentChunkSize, agentWindowSize, and agentFrameSize values), the longer the time that is needed for the agent to recover transfers. For reliable Managed File Transfer networks where transfers rarely enter a recovery state, it may be beneficial to increase this value to increase overall performance.</p>	1

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
agentChunkSize	<p>The size of each transfer chunk for the transport of file data. Hence, denotes the maximum size of the IBM MQ messages that are transferred between the source and the destination agents. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value.</p> <p>This value is negotiated between the source agent and the destination agent, and the larger of the two values is used. If you want to change the value of this property, change the value at both the source agent and at the destination agent.</p> <p>agentChunkSize is an integer value. For example: agentChunkSize = 10240 sets the chunk size to 10 KB.</p>	262144-byte (which is equivalent to 256 KB)
agentFrameSize	<p>The number of windows for the transfer frame. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value.</p> <p>For networks that have high latency, increasing this value may improve overall performance as it causes the agent to have more message chunks active concurrently.</p> <p>The value of this property, multiplied by agentWindowSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer. For example, 262144-byte chunks x 10 x 5 = 12.5 MB for each transfer.</p> <p>Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing this property has no effect on the performance of the transfer.</p>	5
agentWindowSize	<p>The number of chunks for each window. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value.</p> <p>For networks that have high latency, increasing this value may improve overall performance. This is because it causes the agent to have more message chunks active concurrently and reduces the frequency that acknowledgment messages are sent back to the source agent.</p> <p>The value of this property, multiplied by agentFrameSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer, and denotes the upper limit of the IBM MQ message data on the command queue of the destination agent. For example, 262144-byte chunks x 10 x 5 = an upper limit of 12.5 MB, for each transfer.</p> <p>Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing the value of this property has no effect on the performance of the transfer.</p>	10
Code page properties:		
agentCcsid	The code page the agent connects to its agent queue manager with. If you specify a value for agentCcsid, you must also specify a value for agentCcsidName. For information on how to view the known code pages for the JVM, see the -hsc parameter in the fteCreateBridgeAgent command.	1208
agentCcsidName	The Java representation of the agentCcsid. If you specify a value for agentCcsidName, you must also specify a value for agentCcsid.	UTF8
Command properties:		
maxCommandHandlerThreads	Controls the number of threads available for the initial parsing and processing of transfer command messages. When active, the threads require a connection to the queue manager but the threads release the connection when idle.	5
maxCommandOutput	The maximum number of bytes stored for command output. This property applies to commands specified for a managed call and preSource, postSource, preDestination, and postDestination commands for a managed transfer. This limits the length of command output that is written to the transfer log on the SYSTEM.FTE topic.	10240
maxCommandRetries	The maximum number of retries for a command that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	9
maxCommandWait	The maximum wait, in seconds, between retries that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	60

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
immediateShutdownTimeout	<p>For an immediate shutdown of an agent, you can use this property to specify the maximum amount of time in seconds an agent waits for its transfers to complete before forcing a shutdown.</p> <p>Note: Do not change the value of this property to less than the default of 10 seconds. An immediate shutdown of an agent requires sufficient time to end any external processes. If the value of this property is too low, processes might be left running.</p> <p>If the value 0 is specified for this property, the agent waits for all outstanding transfers to stop. If an invalid value is specified for this property, the default value is used.</p>	10
Connection properties:		
javaLibraryPath	When connecting to a queue manager in bindings mode, Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location that is defined by IBM MQ. If the bindings libraries are in a different location, use this property to specify the location of the bindings libraries.	None
Connect:Direct bridge properties:		
cdNode	<p>Required property if you want to use the Connect:Direct bridge.</p> <p>The name of the Connect:Direct node to use to transfer messages from the Connect:Direct bridge agent to destination Connect:Direct nodes. This node is part of the Connect:Direct bridge, not the remote node that is the source or destination of the transfer. For more information, see “The Connect:Direct bridge” on page 220.</p>	No default
cdNodeHost	<p>The host name or IP address of the Connect:Direct node to use to transfer files from the Connect:Direct bridge agent to destination nodes (the Connect:Direct bridge node).</p> <p>In most cases, the Connect:Direct bridge node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct bridge node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct bridge node.</p> <p>If you have not set the cdNode property, this property is ignored.</p>	The host name or IP address of the local system
cdNodePort	<p>The port number of the Connect:Direct bridge node that client applications use to communicate with the node. In Connect:Direct product documentation, this port is referred to as the API port.</p> <p>If you have not set the cdNode property, this property is ignored.</p>	1363
cdTmpDir	<p>The location to store files temporarily on the system where the Connect:Direct bridge agent is running before they are transferred to the destination Connect:Direct node.</p> <p>This property specifies the full path of the directory where files are temporarily stored. For example, if cdTmpDir is set to /tmp then the files are temporarily placed in the /tmp directory.</p> <p>If you have not set the cdNode property, this property is ignored.</p> <p>The value of this property can contain environment variables.</p> <p>See “The use of environment variables in MFT properties” on page 511 for more information.</p>	<p><i>value_of_java.io.tmpdir</i> <i>/cdbridge-agentName</i></p> <p>On Windows,</p> <p><i>value_of_java.io.tmpdir</i> <i>\cdbridge-agentName</i></p>
cdTrace	Whether the agent traces data that is sent between the Connect:Direct bridge agent and its Connect:Direct node. The value of this property can be true or false.	false
cdMaxConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a successful connection has not yet been made, before the transfer fails.	-1 (an infinite number of attempts)
cdMaxPartialWorkConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a previous connection attempt has been successful and transfer work has completed, before the transfer fails.	-1 (an infinite number of attempts)

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
cdMaxWaitForProcessEndStats	The maximum time in milliseconds to wait for Connect:Direct process completion information to become available within the Connect:Direct node statistics information, after the process has ended, before the file transfer is judged to have failed. Typically the information is available immediately, but under certain failure conditions the information is not published. In these conditions the file transfer fails after waiting for the amount of time that is specified by this property.	60000
cdAppName	The application name that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge.	Managed File Transfer <i>current version</i> , where <i>current version</i> is the version number of the product.
cdNodeLocalPortRange	The range of local ports to use for socket connections between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. The format of this value is a comma-separated list of values or ranges. By default, the operating system selects the local port numbers.	None
cdNodeProtocol	The protocol that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge. The following values are valid: <ul style="list-style-type: none"> • TCPIP • SSL • TLS 	TCPIP
cdNodeKeystore	The path to the keystore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. The value of this property can contain environment variables. See “The use of environment variables in MFT properties” on page 511 for more information.	None
cdNodeKeystoreType	The file format of the keystore that is specified by the cdNodeKeystore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks
cdNodeKeystoreCredentialsFile	The path to the file that contains the cdNodeKeystore credentials. The value of this property can contain environment variables. See “The use of environment variables in MFT properties” on page 511 for more information.	 For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42  See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.  Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication
cdNodeTruststore	The path to the truststore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. The value of this property can contain environment variables. See “The use of environment variables in MFT properties” on page 511 for more information.	None
cdNodeTruststoreType	The file format of the truststore that is specified by the cdNodeTruststore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
cdNodeTruststoreCredentialsFile	The path to the file that contains the cdNodeTruststore credentials. The value of this property can contain environment variables. See “The use of environment variables in MFT properties” on page 511 for more information.	See “Configuring MQMFTCredentials.xml on z/OS” on page 42 and “Configuring MQMFTCredentials.xml on multiplatforms” on page 40.  Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication
logCDProcess	The level of Connect:Direct process logging that is recorded in the agent event log in the output0.log file. The values that this property can have are None or Failures or All.	None
File to message and message to file agent properties:		
deleteTmpFileAfterRenameFailure	Setting this property to a value of false ensures that temporary files are not deleted from the destination if the rename operation fails. In this case, the transferred data remains at the destination in a temporary (.part) file. You can manually rename this file later. By default this property has the value of true. This property applies to both message-to-file and file-to-file transfers.	true
enableQueueInputOutput	By default, the agent cannot read data from a source queue or write data to a destination queue as part of a transfer. Setting this value to true enables the agent to perform file to message, and message to file transfers. The value of this property can be true or false.	false
enableSystemQueueInputOutput	Specifies whether the agent can read from or write to IBM MQ system queues. System queues are prefixed with the qualifier SYSTEM. Note: System queues are used by IBM MQ, Managed File Transfer, and other applications to transmit important information. Changing this property enables the agent to access these queues. If you enable this property, use user sandboxing to limit the queues that the agent can access.	false
enableClusterQueueInputOutput	Specifies whether the agent can read from or write to IBM MQ clustered queues. Note: You must specify the enableClusterQueueInputOutput agent property in addition to the enableQueueInputOutput property.	false
maxDelimiterMatchLength	The maximum number of characters that can be matched by the Java regular expression that is used to split a text file into multiple messages as part of a file-to-message transfer.	5
maxInputOutputMessageLength	The maximum length, in bytes, of a message that is read from a source queue or written to a destination queue by an agent. The maxInputOutputMessageLength property of the source agent in a transfer determines how many bytes can be read from a message on the source queue. The maxInputOutputMessageLength property of the destination agent in a transfer determines how many bytes can be written to a message on the destination queue. If the length of the message exceeds the value of this property the transfer fails with an error. This property does not affect the Managed File Transfer internal queues. For information about changing this property, see “Guidance for setting MQ attributes and MFT properties associated with message size” on page 306.	1048576
monitorGroupRetryLimit	The maximum number of times that a monitor triggers a message-to-file transfer again if the message group still exists on the queue. The number of times that the message-to-file transfer triggers is determined from the MQMD backout count of the first message in the group. If the agent is restarted the monitor triggers a transfer again even if the number of times the transfer triggers exceeds the value of monitorGroupRetryLimit. If this behavior causes the number of times that the transfer triggers to exceed the value of monitorGroupRetryLimit, the agent writes an error to its event log. If the value -1 is specified for this property, the monitor triggers the transfer again an unlimited number of times until the trigger condition is not satisfied.	10
General agent properties:		
agentStatusPublishRateLimit	The maximum rate in seconds that the agent republishes its status because of a change in file transfer status. If you set this property to too small a value, the performance of the IBM MQ network might be negatively affected.	30

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
agentStatusPublishRateMin	The minimum rate in seconds that the agent publishes its status. This value must be greater than or equal to the value of the agentStatusPublishRateLimit property.	300
enableMemoryAllocationChecking	<p>Determines whether the Managed File Transfer Agent checks that there is sufficient memory available to run a transfer before a transfer is accepted. The check is made on both the source and destination agents. If there is insufficient memory available, the transfer is rejected.</p> <p>When calculating the memory required for a transfer, the maximum memory that is required by the transfer is used. Therefore, the value might be greater than the actual memory that is used by the transfer. For this reason, the number of concurrent transfers that can run might be reduced if the enableMemoryAllocationChecking property is set to true. You are recommended to set the property to true only if you are experiencing problems with Managed File Transfer failing with out-of-memory errors. The transfers that are likely to consume large amounts of memory are file-to-message and message-to-file transfers where the sizes of the messages are large.</p>	false
enableDetailedReplyMessages	<p>Setting this property to true enables managed transfer request replies to contain detailed information about the transferred files. The detailed information and format is the same as that published to the transfer log in the progress messages, that is, the <transferSet> element. For more information, see “File transfer log message formats” on page 723.</p> <p>The detailed reply information is included only when the managed transfer request specifies that detailed reply information is required. To specify this requirement, set the detailed attribute of the <reply> element of the managedTransfer XML request message sent to the source agent. For more information, see “File transfer request message format” on page 755.</p> <p>Multiple reply messages can be generated for each transfer request. This number is equal to the number of transfer log progress messages for the transfer plus 1 (where the first reply message is a simple ACK reply). Detailed information is included in all messages, except for the ACK reply messages, but the overall transfer result is included only in the last detailed reply message.</p>	true
enableUserMetadataOptions	<p>Determines whether you can use known keys for user-defined metadata in new transfer requests to provide more transfer options. These known keys always start with the following prefix <code>com.ibm.wmqfte.</code>. As a consequence when the enableUserMetadataOptions property is set true, keys that use this prefix are not supported for user-defined use. When the enableUserMetadataOptions property is set to true, the keys that are supported currently are as follows:</p> <ul style="list-style-type: none"> • <code>com.ibm.wmqfte.insertRecordLineSeparator</code> • <code>com.ibm.wmqfte.newRecordOnLineSeparator</code> • <code>com.ibm.wmqfte.convertLineSeparators</code> <p>For information about what these keys mean, see “fteCreateTransfer: start a new file transfer” on page 417.</p> <p>The value of this property can be true or false.</p>	false

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
failTransferOnFirstFailure	<p>Allows an agent to be configured to fail a managed transfer as soon as a transfer item within that managed transfer fails.</p> <p>To enable this feature, APAR IT03450 must be applied for both the source agent and the destination agent, and the failTransferOnFirstFailure property must be set to true in the source agent's agent.properties file. Setting the property to true true on the destination agent is optional.</p> <p>When the failTransferOnFirstFailure property is set to true, the agent starts processing managed transfer requests as normal. However, as soon as a transfer item fails, then the managed transfer is marked as failed and no further transfer items are processed. Transfer items that were successfully processed before the managed transfer failed are handled in the following way:</p> <ul style="list-style-type: none"> • The source disposition for those transfer items is honored. For example, if the source disposition for the transfer item was set to delete, then the source file is deleted. • The destination files that were written remain on the destination file system and are not deleted. <p>If the failTransferOnFirstFailure property is not set to true and a managed file transfer contains multiple files and one of these files fails to transfer, for example because the destination file already exists and the overwrite property is set to error, the source agent continues and attempts to transfer any remaining files in the request.</p>	false
itemsPerProgressMessage	<p>The number of files that are transferred before an agent publishes its next progress log message. Use this property to control the rate that progress log messages are published to the coordination queue manager during a transfer.</p> <p>The maximum value this property can be set to is 1000.</p> <p>Note: Progress messages include information about every file that is transferred since the last progress message was published. Increasing this value increases the size of the progress messages, which might affect performance.</p>	50
maxInlineFileSize	<p>For single file-to-file, or file-to-message transfers, the maximum file size (in bytes) that can be automatically included in the initial transfer request message.</p> <p>You can use this property to improve the speed of your transfers, but if you set the file size to too large a value, this might degrade performance. A suggested initial size for this property is 100 KB but you are recommended to thoroughly test different values until you find the best file size for your system.</p>	0
Input/output properties:		
doNotUseTempOutputFile	<p>By default, the agent writes to a temporary file at the destination and renames this temporary file to the required file name after the file transfer is complete. Setting this value to true causes the agent to write directly to the final destination file.</p> <p> On z/OS systems, this behavior does not apply to sequential data sets, but does apply to PDS data set members.</p> <p>The value of this property for a transfer is defined by the destination agent.</p>	false
enableMandatoryLocking	<p>When accessing normal files, Managed File Transfer takes a shared lock for reading and an exclusive lock for writing.</p> <p> On Windows file locking is advisory only. When this property is set to true, Managed File Transfer enforces file locking. On Windows this means that if another application has a file open, monitoring of that file does not trigger until the file is closed. Managed File Transfer transfers involving that file fail.</p> <p> On UNIX type platforms, file locking is fulfilled across processes. For UNIX type platforms, setting this property has no effect.</p> <p>This property applies to normal Managed File Transfer agents only. Managed File Transfer does not support the file locking mechanism on a file server. This property therefore does not work for a protocol bridge agent because protocol bridge agent does not lock a file on a file server when transferring a file.</p> <p>The value of this property can be true or false.</p>	false
ioIdleThreadTimeout	<p>Time in milliseconds for a file system input/output thread to remain idle before the thread shuts down.</p>	10000

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
ioQueueDepth	The maximum number of input/output requests to queue up.	10
ioThreadPoolSize	<p>Maximum number of file system input/output threads available. Typically each transfer uses its own file system input/output thread, but if the number of concurrent transfers exceeds this limit, the file system input/output threads are shared between transfers.</p> <p>If you think you are likely to regularly have more concurrent transfers in progress than the ioThreadPoolSize value, you might see an improvement by increasing this value, so that each transfer has its own file system input/output thread.</p>	10
textReplacementCharacterSequence	<p>For text mode transfer, if any of the data bytes cannot be converted from the source code page to the destination code page, the default behavior is for the file transfer to fail.</p> <p>Set this property to allow the transfer to complete successfully by inserting the specified character value. This property value is a single character. Typically, a question mark (?) is used for any unmappable characters. For example, use this format textReplacementCharacterSequence=? where the question mark (?) is the replacement character. You cannot use a white space character as a replacement character.</p>	None
Multi-channel support:		
agentMultipleChannelsEnabled	<p>Setting this property to <code>true</code> enables a Managed File Transfer Agent to send transfer data messages across multiple IBM MQ channels. In some scenarios, setting this property might improve performance. However, only enable multi-channel support only if there is a demonstrable performance benefit. Only messages that are put to the <code>SYSTEM.FTE.DATA.destinationAgentName</code> queue are sent across multiple channels. The behavior for all other messages remains unchanged.</p> <p>When you set this property to <code>true</code>, you must also complete the IBM MQ configuration steps in one of the following topics to enable multi-channel support:</p> <ul style="list-style-type: none"> • “Configuring an MFT agent for multiple channels in a cluster” on page 546 • “Configuring an MFT agent for multiple channels: non-clustered” on page 547 <p>Additionally, you must also complete the standard IBM MQ configuration steps that are required for a Managed File Transfer agent, which are detailed in “Configuring MFT for first use” on page 90.</p> <p>The value of this property can be <code>true</code> or <code>false</code>.</p>	false
agentMessageBatchSize	When configured with multiple channels, a source agent sends data messages for a transfer across each channel on a round-robin basis. This property controls the number of messages that are sent down each channel at a time.	5
Multi-instance queue manager properties:		
agentQMGrStandby	<p>The host name and the port number that are used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance agent queue manager that is defined by agentQMGr. For example, <code>host_name(port_number)</code></p> <p>The agent attempts to connect to the standby queue manager when it detects a connection broken error, for example, MQRCA009. Once the agent gets connected to the standby queue manager the agent remains connected until the standby queue manager becomes unavailable.</p>	No default
Process controller properties:		
agentQMGrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the agent's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval that is specified by the value of the maxRestartInterval property. When this value is exceeded the agent's process controller stops restarting the agent, and instead makes an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the agent's process controller measures agent restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the agent's process controller stops restarting the agent. Instead the agent's process controller makes an action that is based on the value of the maxRestartDelay property.	120

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
maxRestartDelay	Determines the behavior of the agent's process controller when the rate of agent restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the agent's process controller is stopped. If you specify a value greater than zero, it is the number of seconds to wait before the restart history information held by the agent's process controller is reset and the agent is restarted.	-1
Protocol bridge properties:		
protocolBridgeCredentialConfiguratio n	The value of this property is passed in as a string to the initialize() method of the exit classes that are specified by protocolBridgeCredentialExitClasses.	null
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see "Mapping credentials for a file server by using exit classes" on page 214.	No default.
protocolBridgeDataTimeout	The timeout in milliseconds that the protocol bridge agent waits to either establish a data connection to an FTP server or to receive data from an FTP server over a connection that is already established. If you set this property to a value of 0, the protocol bridge agent waits indefinitely. If the timeout elapses, the protocol bridge agent closes any existing data connections to the FTP server and attempts to establish a new data connection before resuming the current transfer. If the attempt to establish the new data connection fails, the current transfer also fails.	0
protocolBridgeLogoutBeforeDisconne ct	Specifies whether the protocol bridge agent logs the user out of the file server before closing the FTP session and disconnecting. If you set this property to true, the protocol bridge agent issues an FTP QUIT command to the file server.	false
protocolBridgePropertiesConfiguratio n	Passed as one of the bridge properties to the initialize() method of the exit classes that are specified by the protocolBridgeServerPropertiesExitClasses property.	No default
protocolBridgePropertiesExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge server properties user exit routine. For more information, see "ProtocolBridgePropertiesExit2: Looking up protocol file server properties" on page 209.	No default
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> V 9.0.0 V 9.0.0 </div> <div> Protocol bridge agent logging properties: </div> </div>		
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> V 9.0.0 V 9.0.0 </div> <div> agentLog </div> </div>	Key value pair component and operation to enable or disable logging of FTP commands and responses between the Protocol Bridge Agent and FTP/SFTP/FTPS file servers. For example: agentLog=on Turn on logging for all components agentLog=off Turn off logging for all components agentLog=ftp=on, sftp=on, ftps=off Turn on logging for FTP and SFTP, and turn off for FTPS	No default
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> V 9.0.0 V 9.0.0 </div> <div> agentLogFileSize </div> </div>	Defines the maximum size of a capture file in megabytes. Same as the default for regular trace default file size.	20
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> V 9.0.0 V 9.0.0 </div> <div> agentLogFiles </div> </div>	Defines the maximum number of capture files that are retained before the oldest file is discarded.	10

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
  agentLogFilter	<p>By default captures communication with all FTP servers the agent is connecting to.</p> <p>For example:</p> <ul style="list-style-type: none"> Filter on host/ip address <pre>host=ftpprod.ibm.com, ftp2.ibm.com host=9.182.*</pre> <ul style="list-style-type: none"> Filter based on metadata <pre>metadata="outbound files to xyz corp"</pre>	*
Queue properties:		
publicationMDUser	The MQMD user ID to associate with messages sent to be published by the coordination queue manager. If you do not set this property, the MQMD user ID is set based on the IBM MQ rules for setting MQMD user IDs.	No default
Resource monitoring properties:		
monitorFilepathPlatformSeparator	Specifies whether to use platform-specific path separators within the <code>\$FILEPATH</code> variable. A value of <code>true</code> uses platform-specific path separators. A value of <code>false</code> uses a UNIX style forward slash (<code>/</code>) path separator on all platforms.	true
monitorMaxResourcesInPoll	<p>Specifies the maximum number of monitored resources to be triggered in each poll interval. For example, if you specify a monitor pattern of <code>*.txt</code>, a poll interval of 10 seconds, and set the <code>monitorMaxResourcesInPoll</code> property to 10, the <code>monitorMaxResourcesInPoll</code> property limits the agent to trigger on a maximum of 10 matches for each poll interval. Matching resources beyond the limit of 10 are triggered in later poll intervals.</p> <p>In addition, you can use the <code>monitorMaxResourcesInPoll</code> property in combination with a matching <code>-bs</code> parameter on the <code>fteCreateMonitor</code> command, for example, to restrict each poll interval to triggering one transfer only.</p> <p>A value less than or equal to zero means that the number of monitor resources that are triggered in a polling interval is unlimited.</p>	-1
monitorReportTriggerFail	Specifies whether failure conditions, in the environment and configuration, that are detected in the monitor are reported as a log message to the <code>SYSTEM.FTE</code> topic. A value of <code>true</code> logs messages. A value of <code>false</code> does not log messages.	true
monitorReportTriggerNotSatisfied	Specifies whether a non-satisfied trigger sends a log message to the <code>SYSTEM.FTE</code> topic that contains the details. A value of <code>true</code> logs messages. A value of <code>false</code> does not log messages.	false
monitorReportTriggerSatisfied	Specifies whether a satisfied trigger sends a log message to the <code>SYSTEM.FTE</code> topic that contains the details. A value of <code>true</code> logs messages. A value of <code>false</code> does not log messages.	false
monitorSilenceOnTriggerFailure	The number of consecutive failures of the resource monitor trigger before the failures are no longer reported.	5
monitorStopOnInternalFailure	The number of consecutive internal FFDC conditions of the resource monitor before the monitor changes its state to stop.	10
Root directory properties:		

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
commandPath	<p>Specifies the set of paths that commands can be called by, using one of the following methods:</p> <ul style="list-style-type: none"> Agent Ant call, filecopy, or filemove tasks In an XML message passed to an agent, using one of the supported Managed File Transfer Agent command XML schemas (for example, managedCall or managedTransfer). <p>For information about the valid syntax of the value of the commandPath property, see “commandPath MFT property” on page 347.</p> <p>Important: Take extreme care when you set this property because any command in one of the specified commandPaths can effectively be called from a remote client system that is able to send commands to the agent. For this reason, by default, when you specify a commandPath:</p> <ul style="list-style-type: none"> Any existing agent sandbox is configured by the agent when it starts up so that all commandPath directories are automatically added to the list of directories that have denied access for a transfer. Any existing user sandboxes are updated when the agent starts up so that all the commandPath directories (and their subdirectories) are added as <exclude> elements to the <read> and <write> elements. If the agent is not configured to use either an agent sandbox, or user sandboxes, then a new agent sandbox is created when the agent starts up that has the commandPath directories specified as denied directories. <p>The value of this property can contain environment variables.</p> <p>See “The use of environment variables in MFT properties” on page 511 for more information.</p> <p> You can set the addCommandPathToSandbox property to false to override this default behavior for compatibility with the following releases:</p> <ul style="list-style-type: none"> IBM WebSphere MQ File Transfer Edition. The IBM WebSphere MQ 7.5.0 Fix Pack 1 Managed File Transfer component (or earlier). The IBM WebSphere MQ 7.5.0 Fix Pack 2 Managed File Transfer component (or later) on an installation that does not have the installation property enableFunctionalFixPack=7502 set. <p>Important: Be aware that this override effectively enables a client to transfer any command to the agent system and call the command, and so should be used with extreme care.</p>	None - no commands can be called
  addCommandPathToSandbox	<p>Specifies whether the directories specified by the commandPath property (and all of their subdirectories) should be added to:</p> <ul style="list-style-type: none"> The denied directories for an existing agent sandbox. The <exclude> elements for the <read> and <write> elements for any user sandboxes that have been defined. A new agent sandbox, if an agent has not been configured with either an agent sandbox, or one or more user sandboxes. <p>This provides compatibility with the following releases:</p> <ul style="list-style-type: none"> IBM WebSphere MQ File Transfer Edition. The IBM WebSphere MQ 7.5.0 Fix Pack 1 Managed File Transfer component (or earlier). The IBM WebSphere MQ 7.5.0 Fix Pack 2 Managed File Transfer component (or later) on an installation that does not have the installation property enableFunctionalFixPack=7502 set. <p>For more information, see “commandPath MFT property” on page 347.</p>	True
    additionalWildcardSandboxChecking	<p>Specifies whether additional checks are to be made on wildcard transfers for an agent that has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from.</p> <p>When this property is set to true, the additional checking is enabled. If a transfer request attempts to read a location that is outside of the defined sandbox for file matching of the wildcard, the transfer fails. If there are multiple transfers within one transfer request, and one of these requests fails due to it attempting to read a location outside of the sandbox, the entire transfer fails. If checking fails, the reason for failure is given in an error messages (see “Additional checks for wildcard transfers” on page 49).</p> <p>If the property is omitted or set to false then no additional checks are made on wildcard transfers.</p>	None

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
sandboxRoot	<p>Specifies the set of root paths to include and exclude when you use sandboxing. See “Working with MFT agent sandboxes” on page 45 for information about this feature.</p> <p>Separate paths with a platform-specific path separator. Prefix paths with an exclamation point (!) character to denote paths as excluded from the sandbox. This feature is useful if you want to exclude a subdirectory under an included root path.</p> <p>The sandboxRoot property is not supported on protocol bridge agents.</p> <p>You cannot specify the sandboxRoot property and the userSandboxes property together.</p> <p>The value of this property can contain environment variables.</p> <p>See “The use of environment variables in MFT properties” on page 511 for more information.</p>	None - no sandbox
transferRoot	<p>Default root directory for relative paths that are specified to the agent.</p> <p>The value of this property can contain environment variables.</p> <p>See “The use of environment variables in MFT properties” on page 511 for more information.</p>	The home directory for the user that started the agent process.
transferRootHLQ	Default HLQ (user ID) for non-fully qualified data sets specified to the agent	The user name of the user that started the agent process.
userSandboxes	<p>Restrict the area of the file system that files can be transferred to and from based on the MQMD user name of the user that requests the transfer. For more information, see “Working with MFT user sandboxes” on page 46.</p> <p>The userSandboxes property is not supported on protocol bridge agents.</p> <p>You cannot specify the sandboxRoot property and the userSandboxes property together.</p>	false
Scheduler property:		
maxSchedulerRunDelay	<p>The maximum interval, in minutes, that the agent waits to check for scheduled transfers. Specify a positive integer to enable this property. For more information about why you might want to use this property, see “What to do if your scheduled file transfer does not run or is delayed” on page 299.</p> <p>Because the agent might be reading a command from its command queue at the time that scheduled transfers are due to run, there may be an additional delay before the scheduled transfers are started. In this case, the scheduler runs immediately after that command completes.</p>	-1
Security properties:		
agentQMGrAuthenticationCredentials File	The path to the file that contains the MQ connection credentials.	 For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42.  See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.  Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication
authorityChecking	<p>Specifies whether the security features described in “Restricting user authorities on MFT agent actions” on page 342 are enabled.</p>  From IBM MQ 9.0.0 Fix Pack 3 and IBM MQ 9.0.4, inquire is a required permission on all of the agent authority queues.	false
logAuthorityChecks	The level of authority check logging that is recorded in the agent event log in the output0.log file. The values that this property can have are None or Failures or All.	None

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM MQ. If <i>java</i> is specified, the user name reported by the JVM is flowed as part of the IBM MQ connection request. The values that this property can have are None or <i>java</i> .	None
Timeout properties:		
maxTransferNegotiationTime	<p>The maximum time in milliseconds that a transfer waits for a destination agent to complete negotiation. If negotiation does not complete in this time, the transfer is put into a resynchronization state and allows another transfer, when available, to run.</p> <p>In scenarios where the source or destination agent is under heavy load it is possible that the default value is too low for the agent to respond quickly enough to the negotiation request. This is most likely when a source agent has a large number of resource monitors defined or when its resource monitors are monitoring directories that contain large numbers of files. However, it can also occur when a large number of transfer requests is submitted to an agent. Increasing the value of this property to 200,000 or more may be necessary in such scenarios.</p>	30 000
recoverableTransferRetryInterval	The time to wait in milliseconds between detecting a recoverable transfer error and attempting to resume the transfer.	60 000
senderTransferRetryInterval	The time in milliseconds to wait until a rejected transfer is retried because the destination is already running the maximum number of transfers. Minimum value is 1000.	30 000
transferAckTimeout	<p>Timeout in milliseconds that a transfer waits for acknowledgment or data from the other end before a retry is issued. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value.</p> <p>Acknowledgments are sent from the receiving agent to the sending agent whenever a complete window of data is received. For bandwidth-constrained or unreliable networks and large <i>agentWindowSize</i> and <i>agentChunkSize</i> settings, it is possible that the default is not long enough. This can cause unnecessary retransfer of data between the agents. Therefore increasing this value might be beneficial and may reduce the likelihood of a transfer going into recovery mode because of a slow network.</p>	60 000
transferAckTimeoutRetries	Maximum number of acknowledgment retries for a transfer without a response before the agent gives up and moves the transfer into a recovery state	5
xmlConfigReloadInterval	<p>The interval in seconds between the agent reloading XML configuration files during runtime. To prevent the agent from reloading XML configuration files during runtime set this property to -1. The following XML configuration files are affected by this property:</p> <ul style="list-style-type: none"> • ConnectDirectCredentials.xml • ConnectDirectNodeProperties.xml • ConnectDirectProcessDefinitions.xml • ProtocolBridgeCredentials.xml • ProtocolBridgeProperties.xml • UserSandboxes.xml 	30
Tracing and logging properties:		

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
javaCoreTriggerFile	<p>The full path to a file location that the agent monitors. If the file exists at the specified location the agent startup will trigger a Javacore. After starting the agent, if you update a file at this location, the agent triggers a Javacore file again.</p> <p>A separate thread polls this file every 30 seconds to check whether the file has been created or updated. If the file has been created or updated since the last poll, the agent generates a Javacore file in one of the following directories:</p> <ul style="list-style-type: none"> • UNIX: <code>MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name</code> • Linux: <code>MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name</code> • Windows: <code>MQ_DATA_PATH\mqft\logs\coordination_qmgr_name\agents\agent_name</code> <p>When you specify this property, the agent outputs the following message at startup:</p> <pre>BFGAG0092I The <insert_0> file will be used to request JVM diagnostic information.</pre> <p>The value of this property can contain environment variables.</p> <p>See "The use of environment variables in MFT properties" on page 511 for more information.</p>	None
trace	<p>Trace specification when agent is to be run with trace enabled at agent start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, <code>com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.commandhandler=all</code>. You can specify multiple trace specifications in a colon-separated list. For example, <code>com.ibm.wmqfte.agent.Agent=all:com.ibm.wmqfte.commandhandler=moderate</code>.</p>	None
outputLogFiles	The total number of output .log files to keep. This value applies to an agent's process controller and the agent itself.	5
outputLogSize	The maximum size in MB of each output .log file before output wraps onto the next file. This value applies to an agent's process controller and the agent itself.	1
outputLogEncoding	The character encoding that the agent uses when it writes to the output .log file.	The default character encoding of the platform that the agent is running on.
traceFiles	The total number of trace files to keep. This value applies to an agent's process controller as well and the agent itself.	5
traceSize	The maximum size in MB of each trace file before trace wraps onto the next file. This value applies to an agent's process controller and the agent itself.	20
traceMaxBytes	The limit to the amount of message data that is output in the trace file.	4096 bytes
logTransferRecovery	When this property is set to a value of true, whenever a transfer enters recovery diagnostic events are reported to the agent's event log in the output0.log file.	For IBM MQ 9.0.0.0, the default value is true
logCapture	Captures transfer request messages that are submitted to this agent and log messages that are published by the agent to the coordination queue manager. These captured messages can be helpful when debugging transfer problems. Captured messages are stored in files in the agent log directory called <code>capture?.log</code> . The ? is a numeric value. The file that contains the number 0 holds the newest captured messages.	false
logCaptureFileSize	Defines the maximum size of a capture file in megabytes.	10
logCaptureFiles	Defines the maximum number of capture files that are retained before the oldest file is discarded.	10
logCaptureFilter	A Java regular expression that the agent uses to match the topic name of the message. Only those messages that match the regular expression are captured.	.* (match all)

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
<p>V 9.0.3</p> <p>V 9.0.3</p> <p>resourceMonitorLog</p>	<p>Key value pair of resource monitor and operation to turn on, or turn off, logging.</p> <p>The possible values are:</p> <ul style="list-style-type: none"> • info • moderate • verbose • off <p>For example:</p> <ul style="list-style-type: none"> • resourceMonitorLog=MON1,MON2=info:MON3=off Turn on logging for MON1 and MON2, and turn off logging for MON3. • resourceMonitorLog=info Turn on info level logging for all resource monitors. <p>The resource monitor logs are written to a file named resmoneventN.log, where N stands for a number; for example, resmonevent0.log.</p> <p> Attention: All resource monitors of an agent write to the same log file.</p> <p>See Logging MFT resource monitors for more information.</p>	info
<p>V 9.0.3</p> <p>V 9.0.3</p> <p>resourceMonitorLogFileSize</p>	<p>Defines the maximum size of a capture file in megabytes.</p>	20
<p>V 9.0.3</p> <p>V 9.0.3</p> <p>resourceMonitorLogFiles</p>	<p>Defines the maximum number of capture files that are retained before the oldest file is discarded.</p>	10
Transfer limit properties:		
maxDestinationTransfers	<p>The maximum number of concurrent transfers that the destination agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files.</p> <p>The agent queues transfers when the destination agent reaches the limit that is specified by the maxDestinationTransfers property.</p> <p>If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.agent name), the agent does not start.</p>	<p>25 (for all agents except Connect:Direct)</p> <p>5 (for Connect:Direct bridge agents)</p>
maxFilesForTransfer	<p>The maximum number of transfer items that are allowed for a single managed transfer. If a managed transfer contains more items than the value of maxFilesForTransfer, the managed transfer fails and no transfer items are processed.</p> <p>Setting this property prevents you from accidentally transferring too many files because of a bad transfer request, for example, if a user accidentally specifies the transfer of the root directory / on a UNIX system.</p>	5000
maxSourceTransfers	<p>The maximum number of concurrent transfers that the source agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files.</p> <p>The source agent queues transfers when the destination agent reaches the limit that is specified by the maxSourceTransfers property.</p> <p>If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.agent name), the agent does not start.</p>	<p>25 (for all agents except Connect:Direct bridge agents)</p> <p>5 (for Connect:Direct bridge agents)</p>

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
maxQueuedTransfers	<p>The maximum number of pending transfers that can be queued by a source agent until the agent rejects a new transfer request. You can set this property so that despite of the limits of maxDestinationTransfers and maxSourceTransfers being met or exceeded, any new transfer requests that you make now are accepted, queued and then carried out later.</p> <p>The order that queued transfer requests are processed in is a factor of their priority and how long they have been queued. Old and high priority pending transfers are selected first. Transfers with a low priority that have been on the queue for a long time are selected in preference to newer, higher priority transfers.</p> <p>If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.agent name), the agent does not start.</p>	1000
<p>Transfer recovery timeout properties:</p>		
transferRecoveryTimeout	<p>Set amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer.</p> <p>When the property is not set, the default behavior of the agent is to keep retrying until it successfully recovers the transfer. You can set the following values for the transfer recovery timeout property:</p> <p>-1 The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.</p> <p>0 The agent stops the file transfer as soon as it enters recovery.</p> <p>>0 The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example, transferRecoveryTimeout=21600 indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this parameter is 999999999.</p>	-1
<p>User exit routine properties:</p>		
agentForceConsistentPathDelimiters	Force the path delimiter in the source file and destination file information that is provided to the transfer exits to be the UNIX style: forward slash (/). Valid options are true and false.	false
destinationTransferEndExitClasses	Specifies a comma-separated list of classes that implement a destination transfer user exit routine.	No default
destinationTransferStartExitClasses	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.	No default
exitClassPath	Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines. The agent exits directory is searched before any entries in this class path.	Agent's exits directory
exitNativeLibraryPath	Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.	Agent's exits directory
ioMaxRecordLength	The maximum record length, in bytes, that can be supported for a record-oriented file. Managed File Transfer can support writing to record-oriented files with any record length. However, large record lengths might cause out-of-memory errors, so to avoid these errors the maximum record length is restricted by default to 64 K. When reading from record-oriented files an entire record must fit into a single transfer chunk, therefore the record length is additionally limited by the transfer chunk size. This property is used only for I/O user exit record-oriented files.	64 KB
monitorExitClasses	Specifies a comma-separated list of classes that implement a monitor exit routine. For more information, see "MFT resource monitor user exits" on page 849.	No default
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see "Mapping credentials for a file server by using exit classes" on page 214.	No default.
sourceTransferEndExitClasses	Specifies a comma-separated list of classes that implement a source transfer end exit routine.	No default

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
sourceTransferStartExitClasses	Specifies a comma-separated list of classes that implement a source transfer start exit routine.	No default
IOExitClasses	Specifies a comma-separated list of classes that implement an I/O user exit routine. List only the classes that implement the IOExit interface, that is, do not list classes that implement the other I/O user exit interfaces, for example IOExitResourcePath and IOExitChannel. For more information, see “Using MFT transfer I/O user exits” on page 249 .	No default.
IBM MQ client compression:		
agentDataCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of file data to negotiate with the remote IBM MQ server. You can find information about these compression types in the following topic: Message data compression list . The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE
agentHeaderCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of header data to negotiate with the remote IBM MQ server. Accepted values are MQCOMPRESS_NONE or MQCOMPRESS_SYSTEM. You can find information about these compression types in the following topic: Message header compression list . The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> z/OS </div> <div style="margin-right: 10px;"> z/OS </div> <div> z/OS-specific: </div> </div>		
<div style="display: flex; flex-direction: column; gap: 5px;"> <div style="display: flex; align-items: center;"> V 9.0.2 </div> <div style="display: flex; align-items: center;"> V 9.0.0.1 </div> <div style="display: flex; align-items: center;"> z/OS </div> <div style="display: flex; align-items: center;"> V 9.0.2 </div> <div style="display: flex; align-items: center;"> V 9.0.0.1 </div> <div style="display: flex; align-items: center;"> z/OS </div> </div> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content;">adminGroup</div>	<p>A security manager group. Members of this group can:</p> <ul style="list-style-type: none"> Start the agent by using the fteStartAgent command. Stop the agent by using the fteStopAgent command. Enable or disable trace for the agent by using the fteSetAgentTraceLevel command. Display details of a local agent by running the fteShowAgentDetails command with the -d parameter specified. <p>Define a security manager group, for example MFTADMIN and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the adminGroup property to be the name of this security manager group.</p> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-top: 5px;">adminGroup=MFTADMIN</div>	None
<div style="display: flex; flex-direction: column; gap: 5px;"> <div style="display: flex; align-items: center;"> z/OS </div> <div style="display: flex; align-items: center;"> z/OS </div> </div> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content;">bpxwdynAllocAdditionalOptions</div>	<p>Managed File Transfer uses the BPXWDYN text interface to create and open z/OS data sets. When BPXWDYN is used for data set allocation by default Managed File Transfer ensures, when possible, the data device is mounted (not required for disk-based data sets but is required for tape data sets). Because the options might not be supported for certain environments, use this property to change this behavior. Also when transferring to a data set it is also possible to specify options for BPXWDYN on the command line; these options are in addition to those options specified by this property.</p> <p>Some BPXWDYN options must not be specified when using the bpxwdynAllocAdditionalOptions property in the <code>agent.properties</code> file. For a list of these properties, see “BPXWDYN properties you must not use with MFT” on page 616.</p>	Default is as follows: <ul style="list-style-type: none"> MOUNT for z/OS V1R8 and later
<div style="display: flex; flex-direction: column; gap: 5px;"> <div style="display: flex; align-items: center;"> z/OS </div> <div style="display: flex; align-items: center;"> z/OS </div> </div> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content;">armELEMTYPE</div>	<p>Optional property. If the agent is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMTYPE parameter value specified in the associated ARM policy. For an agent, set ELEMTYPE to SYSBFGAG.</p>	Not set

Table 45. Advanced agent properties (continued)

Property name	Description	Default value
  armELEMENT	Optional property. If the agent is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMENT parameter value specified in the associated ARM policy. You can set the ELEMENT value to correspond to the agent name.	Not set
Other properties:		
  legacyXMLMessageMQMDFormat	Managed File Transfer XML messages that are generated by the agent (for example, log and transfer progress messages), are now sent to a queue with a blank MQMD format field. Previous versions of the product set the MQMD format field to MQSTR (a text message string). Setting this property to true enables the Managed File Transfer XML messages generated by the agent to be sent to a queue with MQMD format field of MQSTR. Note: Agent reply messages to commands will be sent with a message format matching the corresponding command request. If the MQMD format field is set to MQSTR, there is potential for Managed File Transfer command XML messages to be corrupted if there are channels in the MQ network with data conversion enabled.	false
 adjustScheduleTimeForDaylightSav ing	If your enterprise runs scheduled transfers every day, because the scheduled transfer was created with: <ul style="list-style-type: none"> • -oi parameter set to days, and • -tb parameter set to source on the <code>fteCreateTransfer</code> command for example, then setting this property to <i>true</i> will move the scheduled transfer time forward one hour when the clocks go forward by one hour and back by one hour when the clocks go back one hour. For example, if your scheduled transfer is due to run at 1:00 am, when the clocks go forward, the transfer will run at 2:00 am and when the clocks go back, the transfer reverts to 1:00 am.	true

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

[“Timeout option for file transfers in recovery” on page 236](#)

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

[“MFT sandboxes” on page 45](#)

You can restrict the area of the file system that the agent can access as part of a transfer. The area that the agent is restricted to is called the sandbox. You can apply restrictions to either the agent or to the user that requests a transfer.

Related reference

[“Java system properties for MFT” on page 580](#)

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

[“fteCreateAgent \(create an MFT agent\)” on page 371](#)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

[“fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)” on page 375](#)

The **fteCreateBridgeAgent** command creates a Managed File Transfer protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 384](#)

The **fteCreateCDAgent** command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

[“The use of environment variables in MFT properties” on page 511](#)

From IBM WebSphere MQ 7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

Configuring an MFT agent for multiple channels in a cluster

If you want to use IBM MQ multi-channel support in a clustered configuration, first set the `agentMultipleChannelsEnabled` property to `true` and then complete the steps in this topic.

About this task

In a cluster, multi-channel support is enabled by IBM MQ definitions on the queue manager of the destination agent only.

You must complete the steps in this topic in addition to the standard IBM MQ configuration steps required for a Managed File Transfer agent, which are listed in [“Configuring MFT for first use” on page 90](#).

The following configuration examples use **runmqsc** commands.

Procedure

1. Define a cluster-receiver channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
```

where:

- `DESTQMGRNAME` is the name of the queue manager of the destination agent.
- `MFTCLUSTER` is the name of the IBM MQ cluster.

You are recommended to use the `MFTCLUSTER.DESTMGRNAME_n` naming convention for channels, but this convention is not mandatory.

2. Define a queue manager alias corresponding to each channel. For example:

```
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
```

You must use the `SYSTEM.FTE.DESTQMGRNAME_n` naming convention for queue manager aliases because the sending agent searches for queue manager aliases of this format. The numbers that you use for `n` must start at 1 and be consecutive. You must make the definitions cluster-wide so that they are available on the source agent's queue manager.

For both the source agent and destination agent to correctly determine the number of queue manager aliases, do **not** define a default XMITQ for the queue manager.

Related concepts

[“Configuring MFT for first use” on page 90](#)

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Configuring an MFT agent for multiple channels: non-clustered

If you want to use IBM MQ multi-channel support in a non-clustered configuration, first set the `agentMultipleChannelsEnabled` property to `true` and then complete the steps in this topic.

About this task

In a non-clustered configuration, multi-channel support is enabled by IBM MQ definitions on the queue manager of both the source agent and the destination agent.

You must complete the steps in this topic in addition to the standard IBM MQ configuration steps required for a Managed File Transfer agent, which are listed in [“Configuring MFT for first use” on page 90](#).

The following steps assume that sender-receiver channels are being used to communicate between the source and destination queue managers.

The following configuration examples use `runmqsc` commands.

Procedure

1. On the destination agent's queue manager, define a receiver channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(RCVR) TRPTYPE(TCP)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(RCVR) TRPTYPE(TCP)
```

where: `DESTQMGRNAME` is the name of the queue manager of the destination agent.

You are recommended to use the `TO.DESTMGRNAME_n` naming convention for channels, but this convention is not mandatory. The receiver channel names must match the corresponding sender channels on the source agent's queue manager.

2. On the source agent's queue manager, define a transmission queue for each channel that you want to use. For example, if you are using two channels:

```
DEFINE QLOCAL(DESTQMGRNAME_1) USAGE(XMITQ)
DEFINE QLOCAL(DESTQMGRNAME_2) USAGE(XMITQ)
```

You are recommended to use the `DESTMGRNAME_n` naming convention for transmission queues, but this convention is not mandatory. The transmission queues you define are referenced from the sender channel definitions and the queue manager alias definitions in the following steps.

3. On the source agent's queue manager, define a sender channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
XMITQ(DESTQMGRNAME_1)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
XMITQ(DESTQMGRNAME_2)
```

You are recommended to use the `TO.DESTMGRNAME_n` naming convention for the channels, but this convention is not mandatory. The sender channel names must match the corresponding receiver channels on the destination agent's queue manager.

4. On the source agent's queue manager, define a queue manager alias corresponding to each channel. For example:

```

DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_1)
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_2)

```

You must use the SYSTEM.FTE.DESTQMGRNAME_n naming convention for the queue manager aliases because the sending agent searches for queues manager aliases of this format. The numbers that you use for *n* must start at 1 and be consecutive.

For the agent to correctly determine the number of queue manager aliases, do **not** define a default XMITQ for the queue manager.

Related concepts

[“Configuring MFT for first use” on page 90](#)

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Related reference

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

enableUserMetadataOptions: Supported MFT user-defined metadata keys

When the agent property `enableUserMetadataOptions` is set to a value of `true`, the following user-defined metadata keys are supported when specified to a new transfer request.

Key name	Description	Default value
<code>com.ibm.wmqfte.insertRecordLineSeparator</code>	For text transfers. When this key is set to true, specifies that when reading record-oriented files, such as z/OS data sets, line separators are to be inserted between records. When this key is set to false, specifies that when reading record-oriented files, line separators are not to be inserted between records.	true
<code>com.ibm.wmqfte.newRecordOnLineSeparator</code>	For text transfers. When this key is set to true, specifies that when writing to record-oriented files, such as z/OS data sets, that line separators indicate a new record and are not written as part of the data. When this key is set to false, specifies that when writing to record-oriented files that line separators are to be treated like any other character (that is, no record breaks).	true
<code>com.ibm.wmqfte.convertLineSeparators</code>	For text transfers. Specifies whether the line separator sequences CRLF and LF are converted to the required line separator sequence for the destination. This conversion currently only takes effect for the following cases: 1. If the user-defined metadata key <code>com.ibm.wmqfte.newRecordOnLineSeparator</code> is set to false and the transfer is to a record-oriented file. 2. If the user-defined metadata key <code>com.ibm.wmqfte.com.ibm.wmqfte.insertRecordLineSeparator</code> is set to false and the transfer is from a record-oriented file.	true

Related information

[“Table 45” on page 528](#)

[fteCreateTransfer -md parameter](#)

Additional MFT agent configuration files

In addition to the `agent.properties` file, the Managed File Transfer agent can have a number of XML configuration files in its configuration directory.

Configuration files

The following XML configuration files can be used to specify additional information used by the agent:

ProtocolBridgeCredentials.xml

If your agent is a protocol bridge agent, you can use this file to specify the credentials to use to log in to the FTP or SFTP server that the agent connects to.

ProtocolBridgeProperties.xml

If your agent is a protocol bridge agent, you can use this file to define the properties of non-default protocol file servers that the agent connects to. The **fteCreateBridgeAgent** command creates a default protocol file server in this file for you.

ConnectDirectCredentials.xml

If your agent is a Connect:Direct bridge agent, you can use this file to specify the credentials to use to connect to the Connect:Direct nodes involved in a transfer.

ConnectDirectNodeProperties.xml

If your agent is a Connect:Direct bridge agent, you can use this file to specify the operating system information about the Connect:Direct nodes involved in a transfer.

ConnectDirectProcessDefinition.xml

If your agent is a Connect:Direct bridge agent, you can use this file to specify the user-defined Connect:Direct processes to call as part of a file transfer.

UserSandboxes.xml

You can use this file to specify which areas of the file system the agent can read from or write to.

Updating the configuration files

Unlike the `agent.properties` file, you can update the XML configuration files and have the agent pick up the changes without having to restart the agent.

When you submit a transfer, if it has been more than 10 second since the last time the agent checked the XML configuration file, the agent checks the last modified time of the XML configuration file. If the XML configuration file has been modified since the last time the agent read the file, the agent reads the file again. If the contents of the file are valid when compared to the XML schema, the agent updates its information. If the contents of the file are not valid, the agent uses the information from the previous version of the file and writes a message to the `output0.log` file.

Protocol bridge credentials file format

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

The `ProtocolBridgeCredentials.xml` file must conform to the `ProtocolBridgeCredentials.xsd` schema. The `ProtocolBridgeCredentials.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MQMFT installation. Users are responsible for manually creating the `ProtocolBridgeCredentials.xml` file, it is no longer created by the **fteCreateBridgeAgent** command. Sample files are available in the `MQ_INSTALLATION_PATH/mqft/samples` directory of the MQMFT installation.

IBM WebSphere MQ 7.5 introduced a new `<agent>` element that contains the `<server>` or `<serverHost>` element for the named agent.

The `ProtocolBridgeCredentials.xml` file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema - 7.5 or later

The following schema describes which elements are valid in the ProtocolBridgeCredentials.xml file for IBM MQ 8.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeCredentials" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/
ProtocolBridgeCredentials">
  <!--
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
  ProtocolBridgeCredentials.xsd">
    <tns:agent name="agent1">
      <tns:serverHost name="myserver">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:serverHost>
    </tns:agent>

    <tns:agent name="agent2">
      <tns:server name="server*" pattern="wildcard">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:server>
    </tns:agent>

    <tns:agent name="agent3">
      <tns:serverHost name="ftpsServer"
        keyStorePassword="keypass"
        trustStorePassword="trustpass">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
      </tns:serverHost>
    </tns:agent>
  </tns:credentials>
  -->
<element name="credentials" type="tns:credentialsType"/>
<complexType name="credentialsType">
  <sequence>
    <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="agentType">
  <choice minOccurs="0" maxOccurs="1">
    <element name="serverHost" type="tns:serverHostType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="server" type="tns:serverType" minOccurs="0" maxOccurs="unbounded"/>
  </choice>
  <attribute name="name" type="string" use="required"/>
</complexType>
<complexType name="serverHostType">
  <sequence>
    <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="keyStorePassword" type="string" use="optional"/>
  <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
  <attribute name="trustStorePassword" type="string" use="optional"/>
  <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
</complexType>
<complexType name="serverType">
  <sequence>
    <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>

```

```

<attribute name="pattern" type="tns:patternType" use="optional"/>
<attribute name="keyStorePassword" type="string" use="optional"/>
<attribute name="keyStorePasswordCipher" type="string" use="optional"/>
<attribute name="trustStorePassword" type="string" use="optional"/>
<attribute name="trustStorePasswordCipher" type="string" use="optional"/>
</complexType>

<element name="user" type="tns:userType"/>

<complexType name="userType">
  <sequence>
    <element ref="tns:privateKey" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="serverUserId" type="string" use="optional"/>
  <attribute name="serverUserIdCipher" type="string" use="optional"/>
  <attribute name="serverPassword" type="string" use="optional"/>
  <attribute name="serverPasswordCipher" type="string" use="optional"/>
  <attribute name="hostKey" use="optional">
    <simpleType>
      <restriction base="string">
        <pattern
          value="([a-zA-F0-9]){2}(:([a-zA-F0-9]){2})*" />
        </pattern>
      </restriction>
    </simpleType>
  </attribute>
</complexType>

<element name="privateKey" type="tns:privateKeyType"/>

<complexType name="privateKeyType">
  <simpleContent>
    <extension base="string">
      <attribute name="keyPassword" type="string" use="optional"/>
      <attribute name="keyPasswordCipher" type="string" use="optional"/>
      <attribute name="associationName" type="string" use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
-->
Determines the type of pattern matching to use.
-->
<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>
</schema>

```

Understanding the ProtocolBridgeCredentials.xml file

The elements and attributes used in the ProtocolBridgeCredentials.xml file are described in the following list.

<credentials>

Group element containing elements that describe the credentials used by a protocol bridge agent to connect to a protocol server.

<agent>

Element containing a <server> or <serverHost> definition for a named agent.

<server>

The protocol server that the protocol bridge connects to.

The <server> element is not supported for 7.0.4 or earlier.

Attribute	Description
name	The name of the protocol server.

Attribute	Description
pattern	If you have used wildcards or regular expressions to specify the pattern of a protocol server name, use either <code>wildcard</code> or <code>regex</code> .
trustStorePassword or trustStorePasswordCipher	Required when the <code><server></code> element refers to an FTPS server. The password used to access the truststore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<serverHost>

The host name of the protocol server that the protocol bridge connects to.

The `ProtocolBridgeCredentials.xml` file can either contain `<serverHost>` elements or `<server>` elements but you cannot use a mixture of the two different types. When you use `<serverHost>`, the name is matched against the protocol server's host name. When you use `<server>`, the name is matched against the protocol server's name (as defined in the `ProtocolBridgeProperties.xml` file).

Attribute	Description
name	The host name or IP address of the protocol server.
trustStorePassword or trustStorePasswordCipher	Required when the <code><serverHost></code> element refers to an FTPS server. The password used to access the truststore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. This property is optional unless you set the <code>keyStore</code> attribute, in which case it is required. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<user>

A user mapping from a Managed File Transfer user name to a protocol server user name.

Attribute	Description
name	The user name that is used with Managed File Transfer.
serverUserId or serverUserIdCipher	The user name that is used with the protocol server. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
serverPassword or serverPasswordCipher	The password for the user name used on the protocol server. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
hostKey	The server host SSH fingerprint.

<privateKey>

The private key of a user.

Attribute	Description
keyPassword or keyStorePasswordCipher	The password for the private key. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
associationName	A name used for trace and logging.

Protocol bridge properties file format

The ProtocolBridgeProperties.xml file in the agent configuration directory defines properties for protocol file servers.

The ProtocolBridgeProperties.xml file must conform to the ProtocolBridgeProperties.xsd schema. The ProtocolBridgeProperties.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the Managed File Transfer installation. A template file, ProtocolBridgeProperties.xml, is created by the **fteCreateBridgeAgent** command in the agent configuration directory.

The ProtocolBridgeProperties.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema

The following schema describes the ProtocolBridgeProperties.xml file.

Note: The `maxReconnectRetry` and `reconnectWaitPeriod` attributes are not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2, or later.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeProperties" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties">
  <!--
    Example: ProtocolBridgeProperties.xml
  -->
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
      ProtocolBridgeProperties.xsd">
    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml"/>
    <tns:defaultServer name="myserver"/>
    <tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234" platform="windows"
      timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
      listFormat="unix" limitedWrite="false"/>
    <tns:sftpServer name="server1" host="myhost.hursley.ibm.com" platform="windows"
      fileEncoding="UTF-8" limitedWrite="false">
      <limits maxListFileNames="10"/>
    </tns:sftpServer>
  </tns:serverProperties>
  -->

  <!-- Root element for the document -->
  <element name="serverProperties" type="tns:serverPropertiesType"></element>

  <!--
    A container for all protocol bridge server properties
  -->
  <complexType name="serverPropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1"/>
      <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
        <element name="ftpServer" type="tns:ftpServerType"/>
        <element name="sftpServer" type="tns:sftpServerType"/>
        <element name="ftpsServer" type="tns:ftpsServerType"/>
        <element name="ftpsfgServer" type="tns:ftpsfgServerType"/>
        <element name="ftpsfgServer" type="tns:ftpsfgServerType"/>
      </choice>
    </sequence>
  </complexType>
</schema>
```

```

<!--
  A container for a server name
-->
<complexType name="serverName">
  <attribute name="name" type="tns:serverNameType" use="required"/>
</complexType>

<!--
  A container for a credentials file name
-->
<complexType name="credentialsFileName">
  <attribute name="path" type="string" use="required"/>
</complexType>

<!--
  A container for all the information about an FTP server
-->
<complexType name="ftpServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpServerAttributes"/>
  <attribute name="passiveMode" type="boolean" use="optional"/>
</complexType>

<!--
  A container for all the information about an SFG FTP server
-->
<complexType name="ftpsfgServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpServerAttributes"/>
</complexType>

<!--
  A container for all the information about an SFTP server
-->
<complexType name="sftpServerType">
  <sequence>
    <element name="limits" type="tns:sftpLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:sftpServerAttributes"/>
</complexType>

<!--
  A container for all the information about a FTPS server
-->
<complexType name="ftpsServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  A container for all the information about a SFG FTPS server
-->
<complexType name="ftpsfgServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  Attributes common to all server types
-->
<attributeGroup name="generalServerAttributes">
  <attribute name="name" type="tns:serverNameType" use="required"/>
  <attribute name="host" type="string" use="required"/>
  <attribute name="port" type="nonNegativeInteger" use="optional"/>
  <attribute name="platform" type="tns:platformType" use="required"/>
  <attribute name="fileEncoding" type="string" use="required"/>
  <attribute name="limitedWrite" type="boolean" use="optional"/>
  <attribute name="controlEncoding" type="string" use="optional"/>
</attributeGroup>

<!--
  Attributes common to ftp and ftps server types
-->

```

```

<attributeGroup name="ftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes"/>
  <attribute name="timeZone" type="string" use="required"/>
  <attribute name="locale" type="tns:localeType" use="required"/>
  <attribute name="listFormat" type="tns:listFormatType" use="optional"/>
  <attribute name="listFileRecentDateFormat" type="tns:dateFormatType" use="optional"/>
  <attribute name="listFileOldDateFormat" type="tns:dateFormatType" use="optional"/>
  <attribute name="monthShortNames" type="tns:monthShortNamesType" use="optional"/>
</attributeGroup>

<!--
  Attributes common to ftps server types
-->
<attributeGroup name="ftpsServerAttributes">
  <attributeGroup ref="tns:ftpServerAttributes"/>
  <attribute name="ftpsType" type="tns:ftpsTypeType" use="optional"/>
  <attribute name="trustStore" type="string" use="required"/>
  <attribute name="trustStoreType" type="string" use="optional"/>
  <attribute name="keyStore" type="string" use="optional"/>
  <attribute name="keyStoreType" type="string" use="optional"/>
  <attribute name="ccc" type="boolean" use="optional"/>
  <attribute name="protFirst" type="boolean" use="optional"/>
  <attribute name="auth" type="string" use="optional"/>
  <attribute name="connectTimeout" type="nonNegativeInteger" use="optional"/>
</attributeGroup>

<!--
  A container for limit-type attributes for a server. Limit parameters
  are optional, and if not specified a system default will be used.
-->
<complexType name="generalLimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
</complexType>

<complexType name="sftpLimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
  <attribute name="connectionTimeout" type="nonNegativeInteger" use="optional"/>
</complexType>

<!--
  Attributes for limits common to all server types
-->
<attributeGroup name="generalLimitAttributes">
  <attribute name="maxListFileNames" type="positiveInteger" use="optional"/>
  <attribute name="maxListDirectoryLevels" type="nonNegativeInteger" use="optional"/>
  <attribute name="maxReconnectRetry" type="nonNegativeInteger" use="optional"/>
  <attribute name="reconnectWaitPeriod" type="nonNegativeInteger" use="optional"/>
  <attribute name="maxSessions" type="positiveInteger" use="optional"/>
  <attribute name="socketTimeout" type="nonNegativeInteger" use="optional"/>
</attributeGroup>

<!--
  The type for matching valid server names. Server names must be at least 2 characters in length
and
  are limited to alphanumeric characters and the following characters: ".", "_", "/" and "%".
-->
<simpleType name="serverNameType">
  <restriction base="string">
    <pattern value="[0-9a-zA-Z\._/%]{2,}" />
  </restriction>
</simpleType>

<!--
  The types of platform supported.
-->
<simpleType name="platformType">
  <restriction base="string">
    </restriction>
</simpleType>

<!--
  The type for matching a locale specification.
-->
<simpleType name="localeType">
  <restriction base="string">
    <pattern value="(.)[-_](.)/" />
  </restriction>
</simpleType>

<!--
  The types of list format supported (for FTP servers).

```

```

-->
<simpleType name="listFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
  Date format for FTP client directory listing on an FTP server. This is
  the format to be passed to methods setDefaultDateFormatStr and
  setRecentDateFormatStr for Java class:
  org.apache.commons.net.ftp.FTPClientConfig
-->
<simpleType name="dateFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
  A list of language-defined short month names can be specified. These are
  used for translating the directory listing received from the FTP server.
  The format is a string of three character month names separated by "|"
-->
<simpleType name="monthShortNamesType">
  <restriction base="string">
    <pattern value="(...\|){11}{...}" />
  </restriction>
</simpleType>

<!--
  The enumerations of the allowed FTPS types: "implicit" & "explicit"
  If not specified the default is "explicit"
-->
<simpleType name="ftpsTypeType">
  <restriction base="string">
    <enumeration value="explicit" />
    <enumeration value="implicit" />
  </restriction>
</simpleType>

<!--
  Attribute Group for SFTP Servers
-->
<attributeGroup name="sftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes" />
  <attribute name="cipherList" type="string" use="optional" />
</attributeGroup>
</schema>

```

Understanding the ProtocolBridgeProperties.xml file

The elements and attributes that are used in the ProtocolBridgeProperties.xml file are described in the following list:

<serverProperties>

Root element of the XML document

<credentialsFile>

Path to the file containing credentials. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see [“The use of environment variables in MFT properties” on page 511](#)

<defaultServer>

The protocol file server that acts as the default server for file transfers

<ftpServer>

An FTP file server

<sftpServer>

An SFTP file server

<ftpsServer>

An FTPS file server

General server attributes that apply to all types of protocol file server:

Attribute	Description
name	Required. The name of the protocol file server. Protocol server names must be at least two characters in length, are not case-sensitive, and are limited to alphanumeric characters and the following characters: <ul style="list-style-type: none"> • period (.) • underscore (_) • forward slash (/) • percent sign (%)
host	Required. The host name or IP address of the protocol file server that you want to send files to or receive files from.
port	Optional. The port number of the protocol file server that you want to send files to or receive files from.
platform	Required. The platform of the protocol file server that you want to send files to or receive files from. Specify either UNIX or WINDOWS. Set this property according to how you enter paths on your FTP, FTPS, or SFTP server. For example, if you are running an FTP server on Windows but when you log in to the server, you must enter UNIX-style paths (that is, with forward slashes), set this value to UNIX and not WINDOWS. Servers running on Windows often present a UNIX-style file system.
fileEncoding	Required. Defines the character encoding that is used by the file server. This property is used when you transfer files in text mode so that the correct encoding sequences are changed when the files are moved between platforms. For example, UTF-8.
limitedWrite	Optional. The default mode when writing to a file server is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The default is false.
controlEncoding	Optional. The control encoding value for control messages being sent to the protocol file server. This property affects the encoding of the file name that is used and must be compatible with the control encoding of the protocol file server. The default is UTF-8.

General attributes that apply to FTP and FTPS servers only:

Attribute	Description
timeZone	Required. The time zone of the protocol file server that you want to send files to or receive files from. For example: America/New_York or Asia/Tokyo.
locale	Required. The language that is used on the protocol file server that you want to send files to or receive files from. For example: en_US or ja_JP
listFormat	Optional. The listing format that defines the format of the file-listed information that is returned from the protocol file server. Use either Windows or UNIX. The default is UNIX.
listFileRecentDateFormat	Optional. The recent date format (less than a year) for FTP client directory listing on an FTP server. This attribute and the listFileOldDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.

Attribute	Description
listFileOldDateFormat	Optional. The old date format (more than a year) for FTP client directory listing on an FTP server. This attribute and the listFileRecentDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
monthShortNames	Optional. A replacement list of month names that are used to decode date information returned from the protocol file server. This property consists of a list of 12 comma-separated names to override the default locale month values. The default is as defined by the protocol file server.

General attributes that apply to FTP servers only:

Attribute	Description
passiveMode	Optional. Controls whether the connection to the FTP server is passive or active. If you set the value of this property to <code>false</code> , the connection is active. If you set the value to <code>true</code> , the connection is passive. The default is <code>false</code> .

General attributes that apply to FTPS servers only:

Attribute	Description
ftpsType	Optional. Specifies whether the explicit or implicit form of the FTPS protocol is used. The default is <code>explicit</code> .
trustStore	Required. The location of the truststore that is used to determine whether the certificate presented by the FTPS server is trusted.
trustStoreType	Optional. The format of the truststore file. The default is <code>JKS</code> .
keyStore	Optional. The location of the keystore that is used to provide certificate information if challenged by the FTPS server. The default is for the protocol bridge to not be able to connect to FTPS servers that are configured to require the authentication of clients.
keyStoreType	Optional. The format of the keystore file. The default is <code>JKS</code> .
ccc	Optional. Selects whether a clear (unencrypted) command channel is used when authentication has completed. The default value is <code>false</code> , which means that the command channel remains encrypted for the entire duration of the FTPS session. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
protFirst	Optional. Specifies whether the USER/PASS commands are issued to the FTPS server before or after the PBSZ/PROT commands. The default value is <code>false</code> , which means USER/PASS commands are sent first followed by PBSZ/PROT commands. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
auth	Optional. Specifies the protocol that is specified as part of the AUTH command. A specified protocol will be tried first, then the default is to try <code>TLS</code> , <code>SSL</code> , <code>TLS-C</code> , or <code>TLS-P</code> until the FTPS server does not reject with a 504 reply code. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .

<limits>

Container element for attributes that are common to all types of server and for attributes that are specific to a type of server:

General limit attributes that apply to all types of protocol file server:

Attribute	Description
maxListFileNames	Optional. The maximum number of names that are collected when scanning a directory on the protocol file server for file names. The default is 999999999.
maxListDirectoryLevels	Optional. The maximum number of directory levels on the protocol server to recursively scan for file names. The default is 1000.
maxReconnectRetry (This attribute is now deprecated.)	Deprecated. This attribute is not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2, or later. Optional. The maximum number of times a protocol server tries to reconnect before the protocol bridge agent stops trying. The default is 2.
reconnectWaitPeriod (This attribute is now deprecated.)	Deprecated. This attribute is not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2 or later. Optional. The time period, in seconds, to wait to before attempting to reconnect. The default is 10 seconds.
maxSessions	Optional. The maximum number of sessions for the protocol server. This number must be greater than or equal to the sum of the maximum number of source and destination transfers for the protocol bridge agent. The default is the sum of the values for the agent properties maxSourceTransfers, maxDestinationTransfers, and maxCommandHandlerThreads, plus 1. If these three properties are using their default values of 25, 25, and 5, the maxSessions default is then 56.
socketTimeout	Optional. The socket timeout in seconds. The value of this attribute is used during file streaming. The default is 30 seconds.

Limit attribute that applies to SFTP servers only:

Attribute	Description
connectionTimeout	Optional. The time, in seconds, to wait for a response from the protocol file server to a connection request. A timeout indicates that the protocol file server is not available. The default value is 30 seconds.
cipherList	Optional. Specifies a comma-separated list of ciphers that are used to communicate between the protocol bridge agent and the SFTP server. The ciphers are called in the order that they are specified in this list. The cipher must be available on the server and the client before it can be used. The ciphers that the protocol bridge agent supports are as follows: <ul style="list-style-type: none"> • blowfish-cbc • 3des-cbc • aes128-cbc • aes192-cbc • aes256-cbc • aes128-ctr • aes192-ctr • aes256-ctr • 3des-ctr

Attribute	Description
	<ul style="list-style-type: none"> • arcfour • arcfour128 • arcfour256 <p>By default, the list of ciphers used by protocol bridge agents is aes128-cbc , aes192-cbc , aes256-cbc.</p>

Connect:Direct credentials file format

The ConnectDirectCredentials.xml file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

The ConnectDirectCredentials.xml file must conform to the ConnectDirectCredentials.xsd schema. The ConnectDirectCredentials.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MQMFT installation. A sample ConnectDirectCredentials.xml file is located in the `MQ_INSTALLATION_PATH/mqft/samples/credentials` directory of the MQMFT installation.

The file ConnectDirectCredentials.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectCredentials.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  This schema defines the format of the XML file that is located in the agent properties
  directory of a Connect:Direct bridge agent. The XML file ConnectDirectCredentials.xml
  is used by the default credential validation of the Connect:Direct bridge.
  For more information, see the WebSphere MQ InfoCenter
-->
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectCredentials"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
  >
  <!--
    <?xml version="1.0" encoding="UTF-8"?>

    <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
      xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
        ConnectDirectCredentials.xsd">
      <tns:agent name="CDAGENT01">
        <tns:pnode name="cdnode*" pattern="wildcard">
          <tns:user name="MUSR_.*"
            ignorecase="true"
            pattern="regex"
            cdUserId="bob"
            cdPassword="passw0rd"
            pnodeUserId="bill"
            pnodePassword="alacazam">
            <tns:snode name="cdnode2" pattern="wildcard" userId="sue" password="foo"/>
          </tns:user>
        </tns:pnode>
      </tns:agent>
    </tns:credentials>

  -->

  <element name="credentials" type="tns:credentialsType"/>
  </schema>
  </!--
-->
```

```

<complexType name="credentialsType">
  <sequence>
    <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="agentType">
  <sequence>
    <element name="pnode" type="tns:pnodeType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
</complexType>

<complexType name="pnodeType">
  <sequence>
    <element name="user" type="tns:userType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
</complexType>

<complexType name="userType">
  <sequence>
    <element name="snode" type="tns:snodeType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="ignorecase" type="boolean" use="optional"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="cdUserId" type="string" use="optional"/>
  <attribute name="cdUserIdCipher" type="string" use="optional"/>
  <attribute name="cdPassword" type="string" use="optional"/>
  <attribute name="cdPasswordCipher" type="string" use="optional"/>
  <attribute name="pnodeUserId" type="string" use="optional"/>
  <attribute name="pnodeUserIdCipher" type="string" use="optional"/>
  <attribute name="pnodePassword" type="string" use="optional"/>
  <attribute name="pnodePasswordCipher" type="string" use="optional"/>
</complexType>

<complexType name="snodeType">
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="userId" type="string" use="optional"/>
  <attribute name="userIdCipher" type="string" use="optional"/>
  <attribute name="password" type="string" use="optional"/>
  <attribute name="passwordCipher" type="string" use="optional"/>
</complexType>

<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>
</schema>

```

Understanding the ConnectDirectCredentials.xml file

The elements and attributes used in the ConnectDirectCredentials.xml file are described in the following list.

<credentials>

Group element containing elements that describe the credentials used by a Connect:Direct bridge agent to connect to a Connect:Direct node.

<agent>

Group element containing elements for <pnode> definitions for a named agent.

<pnode>

The primary node (PNODE) in the Connect:Direct transfer. This node initiates the connection to the secondary node (SNODE).

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.

Attribute	Description
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> • wildcard - wildcards are used • regex - Java regular expressions are used

<user>

The IBM MQ user that submits the transfer request.

Attribute	Description
name	The user name that is used with Managed File Transfer. The value of this attribute can be a pattern that matches many user names.
ignorecase	Specifies whether the case of the name is ignored. Valid values for the ignorecase attribute are <ul style="list-style-type: none"> • true - the name is not case sensitive • false - the name is case sensitive
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> • wildcard - wildcards are used • regex - Java regular expressions are used
cdUserId or cdUserIdCipher	The user name that is used by the Connect:Direct bridge to connect to its associated Connect:Direct node. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
cdPassword or cdPasswordCipher	The password associated with the user name specified by the cdUserId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
pnodeUserId or pnodeUserIdCipher	The user name that is used by the Connect:Direct primary node. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
pnodePassword or pnodePasswordCipher	The password associated with the user name specified by the pnodeUserId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<snode>

The Connect:Direct node that performs the role of secondary node (SNODE) during the Connect:Direct file transfer.

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> • wildcard - wildcards are used • regex - Java regular expressions are used

Attribute	Description
userId or userIdCipher	The user name used to connect to this node during a file transfer. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
password or passwordCipher	The password associated with the user name specified by the userId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

Example

In this example, the Connect:Direct bridge agent connects to the Connect:Direct node pnode1. When an IBM MQ user with the user name beginning with the prefix fteuser followed by a single character, for example fteuser2, requests a transfer involving the Connect:Direct bridge, the Connect:Direct bridge agent will use the user name cduser and the password passw0rd to connect to the Connect:Direct node pnode1. When the Connect:Direct node pnode1 performs its part of the transfer it uses the user name pnodeuser and the password passw0rd1.

If the secondary node in the Connect:Direct transfer has a name that begins with the prefix FISH, the node pnode1 uses the user name fishuser and the password passw0rd2 to connect to the secondary node. If the secondary node in the Connect:Direct transfer has a name that begins with the prefix CHIPS, the node pnode1 uses the user name chipsuser and the password passw0rd3 to connect to the secondary node.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
ConnectDirectCredentials.xsd">
  <tns:agent name="CDAGENT01">
    <tns:pnode name="pnode1" pattern="wildcard">
      <tns:user name="fteuser?" pattern="wildcard" ignorecase="true"
        cdUserId="cduser" cdPassword="passw0rd"
        pnodeUserId="pnodeuser" pnodePassword="passw0rd1">
      <tns:snode name="FISH*" pattern="wildcard"
        userId="fishuser" password="passw0rd2"/>
      <tns:snode name="CHIPS*" pattern="wildcard"
        userId="chipsuser" password="passw0rd3"/>
    </tns:user>
  </tns:pnode>
</tns:agent>
</tns:credentials>
```

Connect:Direct node properties file format

The ConnectDirectNodeProperties.xml file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

The ConnectDirectNodeProperties.xml file must conform to the ConnectDirectNodeProperties.xsd schema. The ConnectDirectNodeProperties.xsd schema document is located in the *MQ_INSTALLATION_PATH/mqft/samples/schema* directory of the MFT installation. A template ConnectDirectNodeProperties.xml file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file ConnectDirectNodeProperties.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the `ConnectDirectNodeProperties.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties">
  <element name="nodeProperties" type="tns:nodePropertiesType"></element>
  <complexType name="nodePropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1"/>
      <element name="node" type="tns:nodeType" minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>
  <complexType name="nodeType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="type" type="string" use="required"/>
  </complexType>
  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex"/>
      <enumeration value="wildcard"/>
    </restriction>
  </simpleType>
</schema>
```

Understanding the `ConnectDirectNodeProperties.xml` file

The elements and attributes used in the `ConnectDirectNodeProperties.xml` file are described in the following list.

nodeProperties

Root element of the XML document.

credentialsFile

Path to the credentials file where sensitive information is stored. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see [“The use of environment variables in MFT properties” on page 511](#)

node

Specifies one or more Connect:Direct nodes.

Attribute	Description
name	A pattern that identifies the names of Connect:Direct nodes that use the definitions specified by the node element. Pattern matching is not case sensitive.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are: <ul style="list-style-type: none">wildcard - wildcards are usedregex - Java regular expressions are used For information about the types of regular expressions used by MFT, see “Regular expressions used by MFT” on page 633 .
type	Specifies the operating system type of the Connect:Direct node or nodes that match the pattern given by the name attribute. Valid values for the type attribute are: <ul style="list-style-type: none">Windows - the node runs on Windows

Attribute	Description
	<ul style="list-style-type: none"> • UNIX - the node runs on UNIX or Linux •  z/OS, zos, os/390, or os390 - the node runs on z/OS <p>The value of this attribute is not case sensitive.</p>

Example

In this example, the Connect:Direct credentials filename is specified as `ConnectDirectCredentials.xml`. The example code specifies the following platform connections:

- All Connect:Direct nodes that have a name that begins with "cdnodew" run on the Windows platform.
- All Connect:Direct nodes that have a name that begins with "cdnodeu" run on the UNIX platform.
- All Connect:Direct nodes that have a name that begins with "cdnodez" run on the z/OS platform.
- All other Connect:Direct nodes run on the UNIX platform.

The Connect:Direct bridge agent searches for matches from the start of the file to the end, and uses the first match that it finds.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:nodeProperties xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectNodeProperties
    ConnectDirectNodeProperties.xsd">

  <tns:credentialsFile path="ConnectDirectCredentials.xml"/>
  <tns:node name="cdnodew*" pattern="wildcard" type="windows"/>
  <tns:node name="cdnodeu.*" pattern="regex" type="unix"/>
  <tns:node name="cdnodez*" pattern="wildcard" type="zos"
  <tns:node name="*" pattern="wildcard" type="unix"/>

</tns:nodeProperties>
```

Connect:Direct process definitions file format

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

The `ConnectDirectProcessDefinitions.xml` file must conform to the `ConnectDirectProcessDefinitions.xsd` schema. The `ConnectDirectProcessDefinitions.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MFT installation. A template `ConnectDirectProcessDefinitions.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectProcessDefinitions.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema

The following schema describes which elements are valid in the `ConnectDirectProcessDefinitions.xml` file.

```
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions">

  <element name="cdprocess" type="tns:cdprocessType"></element>

  <complexType name="cdprocessType">
```

```

    <sequence>
      <element name="processSet" type="tns:processSetType"
        minOccurs="0" maxOccurs="unbounded"/></element>
    </sequence>
  </complexType>

  <complexType name="processSetType">
    <sequence>
      <element name="condition" type="tns:conditionType"
        minOccurs="0" maxOccurs="1"/>
      <element name="process" type="tns:processType"
        minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>

  <complexType name="conditionType">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element name="match" type="tns:matchType"/>
      <element name="defined" type="tns:definedType"/>
    </choice>
  </complexType>

  <complexType name="matchType">
    <attribute name="variable" type="string" use="required"/>
    <attribute name="value" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
  </complexType>

  <complexType name="definedType">
    <attribute name="variable" type="string" use="required"/>
  </complexType>

  <complexType name="processType">
    <sequence>
      <element name="preTransfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="transfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="postTransferSuccess" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="postTransferFailure" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>

  <complexType name="transferType">
    <attribute name="process" type="string" use="required"/>
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex"/>
      <enumeration value="wildcard"/>
    </restriction>
  </simpleType>
</schema>

```

Understanding the ConnectDirectProcessDefinitions.xml file

The elements and attributes used in the ConnectDirectProcessDefinitions.xml file are described in the following list.

cdProcess

The root element of the XML document.

processSet

Group element containing all the information about a set of user-defined processes.

condition

Group element containing the conditions that a transfer is tested against to determine whether the set of processes contained in the processSet element are used.

match

A condition that tests whether a the value of a variable matches a given value.

Attribute	Description
variable	Specifies a variable. The value of this variable is compared with the value of the value attribute. The variable is an intrinsic symbol. For more information, see “Substitution variables for use with user-defined Connect:Direct processes” on page 633.
value	Specifies a pattern to match against the value of the variable specified by the variable attribute.
pattern	Specifies the type of pattern that is used for the value of the value attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> • wildcard - wildcards are used • regex - Java regular expressions are used This attribute is optional and the default is <code>wildcard</code> .

defined

A condition that tests whether a variable has been defined.

Attribute	Description
variable	Specifies a variable. If this variable exists, the match condition is satisfied. The variable is an intrinsic symbol. For more information, see “Substitution variables for use with user-defined Connect:Direct processes” on page 633.

process

Group element containing the information about where to locate the Connect:Direct processes to call when a match is found.

transfer

The Connect:Direct process to call during a transfer request.

Attribute	Description
process	Optional. Specifies the name of a file that contains a Connect:Direct process to call during a transfer request. The file path is relative to the Connect:Direct bridge agent configuration directory. This attribute is optional, the default is to use a process generated by MFT. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see “The use of environment variables in MFT properties” on page 511

Example

In this example, there are three processSet elements.

The first processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*` and a **%FTESUSER** variable with a value of `Admin`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/AdminClient.cdp` as part of the transfer.

The second processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/Client.cdp` as part of the transfer. The Connect:Direct bridge agent reads the processSet elements in the order that they are defined, and if it finds a match, it uses the first match and does not look for another match. For transfer requests that match the conditions of both the first and second processSet, the Connect:Direct bridge agent calls only the processes specified by the first processSet.

The third processSet element has no conditions and matches all transfers. If the transfer request does not match the conditions of the first or second processSet, the Connect:Direct bridge agent

submits the Connect:Direct process specified by the third condition. This process is located in the `agent_configuration_directory/Default.cdp` as part of the transfer.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions
ConnectDirectProcessDefinitions.xsd">
  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard"/>
      <tns:match variable="%FTESUSER" value="Admin" pattern="wildcard"/>
    </tns:condition>
    <tns:process>
      <tns:transfer process="AdminClient.cdp"/>
    </tns:process>
  </tns:processSet>

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard"/>
    </tns:condition>
    <tns:process>
      <tns:transfer process="Client.cdp"/>
    </tns:process>
  </tns:processSet>

  <tns:processSet>
    <tns:process>
      <tns:transfer process="Default.cdp"/>
    </tns:process>
  </tns:processSet>
</tns:cdprocess>
```

Working with MFT user sandboxes

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

User sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent.

To enable user sandboxing, add the following property to the agent `.properties` file for the agent that you want to restrict:

```
userSandboxes=true
```

When this property is present and set to true the agent uses the information in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/UserSandboxes.xml` file to determine which parts of the file system the user who requests the transfer can access.

The `UserSandboxes.xml` XML is composed of an `<agent>` element that contains zero or more `<sandbox>` elements. These elements describe which rules are applied to which users. The `user` attribute of the `<sandbox>` element is a pattern that is used to match against the MQMD user of the request.

The file `UserSandboxes.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

If you specify the `userPattern="regex"` attribute or value, the `user` attribute is interpreted as a Java regular expression. For more information, see [“Regular expressions used by MFT” on page 633](#).

If you do not specify the `userPattern="regex"` attribute or value the `user` attribute is interpreted as a pattern with the following wildcard characters:

- asterisk (*), which represents zero or more characters
- question mark (?), which represents exactly one character

Matches are performed in the order that the <sandbox> elements are listed in the file. Only the first match is used, all following potential matches in the file are ignored. If none of the <sandbox> elements specified in the file match the MQMD user associated with the transfer request message, the transfer cannot access the file system. When a match has been found between the MQMD user name and a user attribute, the match identifies a set of rules inside a <sandbox> element that are applied to the transfer. This set of rules is used to determine which files, or data sets, can be read from or written to as part of the transfer.

Each set of rules can specify a <read> element, which identifies which files can be read, and a <write> element which identifies which files can be written. If you omit the <read> or <write> elements from a set of rules, it is assumed that the user associated with that set of rules is not allowed to perform any reads or any writes, as appropriate.

Note: The <read> element must be before the <write> element, and the <include> element must be before the <exclude> element, in the UserSandboxes.xml file.

Each <read> or <write> element contains one or more patterns that are used to determine whether a file is in the sandbox and can be transferred. Specify these patterns by using the <include> and <exclude> elements. The name attribute of the <include> or <exclude> element specifies the pattern to be matched. An optional type attribute specifies whether the name value is a file or queue pattern. If the type attribute is not specified, the agent treats the pattern as a file or directory path pattern. For example:

```
<tns:read>
  <tns:include name="/home/user/**"/>
  <tns:include name="USER.**" type="queue"/>
  <tns:exclude name="/home/user/private/**"/>
</tns:read>
```

The <include> and <exclude> name patterns are used by the agent to determine whether files, data sets, or queues can be read from or written to. An operation is allowed if the canonical file path, data set, or queue name matches at least one of the included patterns and exactly zero of the excluded patterns. The patterns specified by using the name attribute of the <include> and <exclude> elements use the path separators and conventions appropriate to the platform that the agent is running on. If you specify relative file paths, the paths are resolved relative to the transferRoot property of the agent.

When specifying a queue restriction, a syntax of QUEUE@QUEUEMANAGER is supported, with the following rules:

- If the at character (@) is missing from the entry, the pattern is treated as a queue name that can be accessed on any queue manager. For example, if the pattern is name it is treated the same way as name@**.
- If the at character (@) is the first character in the entry, the pattern is treated as a queue manager name and all queues on the queue manager can be accessed. For example, if the pattern is @name it is treated the same way as **@name..

The following wildcard characters have special meaning when you specify them as part of the name attribute of the <include> and <exclude> elements:

A single asterisk matches zero or more characters in a directory name, or in a qualifier of a data set name or queue name.

?

A question mark matches exactly one character in a directory name, or in a qualifier of a data set name or queue name.

Two asterisk characters match zero or more directory names, or zero or more qualifiers in a data set name or queue name. Also, paths that end with a path separator have an implicit "***" added to the end of the path. So /home/user/ is the same as /home/user/**.

For example:

- `/**/test/**` matches any file that has a `test` directory in its path
- `/test/file?` matches any file inside the `/test` directory that starts with the string `file` followed by any single character
- `c:\test*.txt` matches any file inside the `c:\test` directory with a `.txt` extension
- `c:\test***.txt` matches any file inside the `c:\test` directory, or one of its subdirectories that has a `.txt` extension
-  `// 'TEST.*.DATA'` matches any data set that has the first qualifier of `TEST`, has any second qualifier, and a third qualifier of `DATA`.
- `*@QM1` matches any queue on the queue manager `QM1` that has a single qualifier.
- `TEST.*.QUEUE@QM1` matches any queue on the queue manager `QM1` that has the first qualifier of `TEST`, has any second qualifier, and a third qualifier of `QUEUE`.
- `**@QM1` matches any queue on the queue manager `QM1`.

Symbolic links

You must fully resolve any symbolic links that you use in file paths in the `UserSandboxes.xml` file by specifying hard links in the `<include>` and `<exclude>` elements. For example, if you have a symbolic link where `/var` maps to `/SYSTEM/var`, you must specify this path as `<tns:include name="/SYSTEM/var"/>`, otherwise the intended transfer fails with a user sandbox security error.

Example

This example shows how to allow the user with the MQMD user name `guest` to transfer any file from the `/home/user/public` directory or any of its subdirectories on the system where the agent `AGENT_JUPITER` is running, by adding the following `<sandbox>` element to the file `UserSandboxes.xml` in `AGENT_JUPITER`'s configuration directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
  <tns:agent>
    <tns:sandbox user="guest">
      <tns:read>
        <tns:include name="/home/user/public/**"/>
      </tns:read>
    </tns:sandbox>
  </tns:agent>
</tns:userSandboxes>
```

Example

This example shows how to allow any user with the MQMD user name `account` followed by a single digit, for example `account4`, to complete the following actions:

- Transfer any file from the `/home/account` directory or any of its subdirectories, excluding the `/home/account/private` directory on the system where the agent `AGENT_SATURN` is running
- Transfer any file to the `/home/account/output` directory or any of its subdirectories on the system where the agent `AGENT_SATURN` is running
- Read messages from queues on the local queue manager starting with the prefix `ACCOUNT.` unless it starts with `ACCOUNT.PRIVATE.` (that is has `PRIVATE` at the second level).
- Transfer data onto queues starting with the prefix `ACCOUNT.OUTPUT.` on any queue manager.

To allow a user with the MQMD user name account to complete these actions, add the following <sandbox> element to the file UserSandboxes.xml, in AGENT_SATURN's configuration directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
  <tns:agent>
    <tns:sandbox user="account[0-9]" userPattern="regex">
      <tns:read>
        <tns:include name="/home/account/**"/>
        <tns:include name="ACCOUNT.**" type="queue"/>
        <tns:exclude name="ACCOUNT.PRIVATE.**" type="queue"/>
        <tns:exclude name="/home/account/private/**"/>
      </tns:read>
      <tns:write>
        <tns:include name="/home/account/output/**"/>
        <tns:include name="ACCOUNT.OUTPUT.**" type="queue"/>
      </tns:write>
    </tns:sandbox>
  </tns:agent>
</tns:userSandboxes>
```

Related reference

[“Additional checks for wildcard transfers” on page 49](#)

From IBM MQ 9.0.1 in Continuous Delivery, and IBM MQ 9.0.0 Fix Pack 1 in Long Term Support, if an agent has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from, you can specify that additional checks are to be made on wildcard transfers for that agent.

[“Working with MFT user sandboxes” on page 46](#)

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

[“The MFT.agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, agent.properties, that must contain the information that an agent uses to connect to its queue manager. The agent.properties file can also contain properties that alter the behavior of the agent.

MFT logger configuration properties

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the logger.properties file, which is in the MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name directory.

For IBM WebSphere MQ 7.5, or later, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see [“The use of environment variables in MFT properties” on page 511](#).

Note: When you specify file paths on Windows, the backslash (\) separator character must appear as double backslashes (\\) (that is, escaped \). Alternatively, you can use a single forward slash character (/) as a separator. For more information about character escaping in Java properties files in Oracle, see [Javadoc for the Properties class](#).

Bindings mode connection properties

Property name	Description	Default value
wmqfte.logger.type	The logger type in use: file, or database. Set this value to FILE, or DATABASE.	No default value

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.max.transaction.messages	The maximum number of messages that is processed in a transaction before the transaction is committed. In circular logging mode, a queue manager has a fixed amount of space available for inflight data. Ensure that you set this property with a sufficiently low value so that the available space does not run out.	50
wmqfte.max.transaction.time	The maximum length of time in milliseconds that passes between transaction commits.	5000
wmqfte.max.consecutive.reject	The maximum number of messages that can be rejected consecutively (that is, without encountering a valid message). If this number is exceeded the logger concludes that the problem is not with the messages themselves but with the configuration. For example, if you make an agent-name column in the database narrower than all of your agent names, all messages referring to agents are rejected.	50
wmqfte.reject.queue.name	The name of a queue to which the logger puts messages that the logger cannot handle. If you have a database logger see Database logger error handling and rejection for details of which messages might be put onto this queue.	SYSTEM.FTE.LOG.RJCT. <i>logger_name</i>
wmqfte.command.queue.name	The name of a queue that the logger reads command messages controlling its behavior from.	SYSTEM.FTE.LOG.CMD. <i>logger_name</i>
wmqfte.queue.manager	The queue manager that the logger connects to. This parameter is required, and is all that is needed for bindings mode connections to the queue manager. (For the properties for connecting to a remote queue manager, see Table 48 on page 578.)	No default value
wmqfte.message.source.type	One of the following values: automatic subscription The default value. The logger creates and uses its own durable, managed subscription on the queue manager that is defined in SYSTEM.FTE/Log/#. This is an appropriate value for most scenarios. administrative subscription If the automatic subscription is not appropriate, you can define a different subscription (for example, by using the IBM MQ Explorer, MQSC, or PCF) and instruct the logger to use that subscription. For example, use this value to partition the log space so that one logger handles agents from A-H, another logger handles I-P, and a third logger from Q-Z. queue If the IBM MQ topology means that creating a subscription for the logger is not convenient, you can use a queue instead. Configure IBM MQ so that the queue receives the messages that are typically received by a subscription to SYSTEM.FTE/Log/# on the coordination queue manager.	automatic subscription
wmqfte.message.source.name	If the message source type is administrative subscription or queue, the name of the subscription or queue to use. This property is ignored if the source type is automatic subscription.	No default value

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
<p>wmqfte.database.credentials.file</p>	<p>The file that contains the user name and password for connecting to the database.</p> <p>For IBM WebSphere MQ 7.5, or later, the value of this property can contain environment variables.</p> <p>For more information, see “MFT credentials file format” on page 785.</p>	<p>z/OS For details on creating the Authentication Credentials File see “Configuring MQMFTCredentials.xml on z/OS” on page 42</p> <p>ULW See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.</p> <p>ULW Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication</p>
<p>wmqfte.database.driver</p>	<p>The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file. For example, the Type 2 driver for Db2 on AIX systems requires the file <code>/opt/IBM/db2/V9.5/java/db2jcc.jar</code>. On Windows systems, specify the path separator as a forward slash character (/) for example, <code>C:/Program Files/IBM/SQLLIB/java/db2jcc.jar</code>.</p> <p>z/OS On z/OS, specify the full path of the <code>db2jcc.jar</code> file. For example, <code>wmqfte.database.driver=/db2/db2v10/jdbc/classes/db2jcc.jar</code>.</p> <p>z/OS On z/OS systems, you must reference all of the following JAR files:</p> <ul style="list-style-type: none"> • <code>db2jcc.jar</code> • <code>db2jcc_license_cisuz.jar</code> • <code>db2jcc_javax.jar</code> <p>If your database driver consists of multiple JAR files (for example, Db2 9.1 requires a driver JAR file and a license JAR file), include all of these JAR files in this property. Separate multiple file names by using the classpath separator for your platform, that is, the semicolon character (;) on Windows systems and the colon character (:) on other platforms.</p>	<p>No default value</p>

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.database.exclude_duplicate_metadata	<p>Controls whether entries are stored in the metadata table that contains information that can be found in other tables within the database logger schema. Set this value to <code>true</code>, or <code>false</code>. These metadata entries are no longer stored by default as it is duplication of existing data and a waste of database storage capacity. The property entries and the tables, where the same data appears, are as follows:</p> <ul style="list-style-type: none"> • <code>com.ibm.wmqfte.SourceAgent TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.DestinationAgent TRANSFER_EVENT</code> • <code>com.ibm.wmqfte.MqmdUser TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.OriginatingUser TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.OriginatingHost TRANSFER_EVENT</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.TransferId TRANSFER</code> or <code>CALL_REQUEST</code> • <code>com.ibm.wmqfte.JobName TRANSFER</code> or <code>CALL_REQUEST</code> <p>Setting the value of this property to <code>false</code> causes these metadata entries to be stored in the metadata table.</p>	true
wmqfte.database.host	<p>Db2 only:</p> <p>For IBM WebSphere MQ 7.5, or later, the host name of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for <code>wmqfte.database.port</code> must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver.</p> <p>If a value for this property is specified, then a credentials file for this logger (file path defined by the <code>wmqfte.database.credentials.file</code> property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.</p>	No default value
wmqfte.database.name	<p>The name of the database instance (or subsystem when using Db2 for z/OS) that contains the Managed File Transfer log tables.</p>	No default value
wmqfte.database.type	<p>The database management system in use: Db2 or Oracle. Set this value to <code>db2</code> or <code>oracle</code>.</p>	db2
wmqfte.database.port	<p>Db2 only:</p> <p>For IBM WebSphere MQ 7.5, or later, the port number of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for <code>wmqfte.database.host</code> must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver.</p> <p>If a value for this property is specified, then a credentials file for this logger (file path defined by the <code>wmqfte.database.credentials.file</code> property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.</p>	No default value

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.database.schema	Db2 only: The database schema that contains the Managed File Transfer logging tables. In most cases the default value is appropriate, but you might need to specify an alternative value depending on your own site-specific database considerations.	FTELOG
wmqfte.database.native.library.path	The path that contains the native libraries that are needed by your chosen database driver (if any). For example, the Type 2 driver for Db2 on AIX systems requires libraries from /opt/IBM/db2/V9.5/lib32/. As an alternative to this property, you can set the java.library.path system property by using other methods. On Solaris and HP-UX systems, before running the fteStartLogger command, you must also set and export the LD_LIBRARY_PATH environment variable to include the path.	No default value
wmqfte.file.logger.fileDirectory	The directory where the file logger log files are located.	mqft/logs/coordination_dir/loggers/logger_name/logs
wmqfte.file.logger.fileSize	The maximum size that a log file is allowed to grow to. The size value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example, <code>wmqfte.file.logger.fileSize=5MB</code> Specifies a maximum file size of 5MB. <code>wmqfte.file.logger.fileSize=2d</code> Specifies a maximum file size of 2 days of data.	10MB
wmqfte.file.logger.fileCount	The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of files never exceeds the value that is specified.	3

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.file.logger.mode	<p>The logger mode in use: circular, or linear. Set this value to CIRCULAR, or LINEAR.</p> <p>CIRCULAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the wmqfte.file.logger.fileSize property. When the maximum size is reached, the file logger starts a new file. The maximum number of files that are written in this mode is controlled by the value that is defined by using the wmqfte.file.logger.fileCount property. When this maximum number of files is reached, the file logger deletes the first file and re-creates it for use as the currently active file. If the value defined in the wmqfte.file.logger.fileSize property is a fixed size byte unit (for example, KB, MB, or GB) then the upper limit on disk space that is used in this mode equals fileSize multiplied by fileCount. If the value defined in the wmqfte.file.logger.fileSize property is a time unit (for example, m, h, d, or w) then the maximum size depends on the throughput of log messages in your system over these time periods. The log file naming convention that is used when running in this mode is: <i>logger_name</i>number-<i>timestamp</i>.log where:</p> <ul style="list-style-type: none"> • <i>logger_name</i> is the name that is given to the logger in the fteCreateLogger command. • <i>number</i> is the number of the file within the set. • <i>timestamp</i> is the timestamp of when the file was created. <p>For example, LOGGER1-20111216123430147.log</p> <p>LINEAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the wmqfte.file.logger.fileSize property. When the maximum size is reached the file logger starts a new file. Previously written files are not deleted, which allows them to be kept as a historical record of log messages. Files are not deleted when running in linear mode, so the wmqfte.file.logger.fileCount property is ignored because there is no upper limit to the number of files that can be created. Because there is no upper limit when running in this mode, it is necessary to track the amount of disk space that is used by the log files to avoid running low on disk space. The log file naming convention that is used when running in this mode is: <i>logger_name</i>-<i>timestamp</i>.log where:</p> <ul style="list-style-type: none"> • <i>logger_name</i> is the name that is given to the logger in the fteCreateLogger command. • <i>timestamp</i> is the timestamp of when the file was created. <p>For example, LOGGER-20111216123430147.log</p>	No default value

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
wmqfte.max.retry.interval	<p>The maximum time, in seconds, between retries when the logger encounters a persistent error.</p> <p>Some error conditions (for example, loss of database connection) prevent the logger from continuing. When this type of condition occurs, the logger rolls back the current transaction, waits for a period, and then retries. The time that the logger waits is initially very short, so that transitory errors can be overcome quickly. However, each time the logger retries, the time that it waits is increased. This prevents too much unnecessary work from taking place when the error condition is longer-lasting, for example when a database is taken down for maintenance.</p> <p>Use this property to set a limit to the length of the wait so that a retry occurs in a reasonable time of the error condition being resolved.</p>	600
loggerQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the logger's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval specified by the value of the maxRestartInterval property. When this value is exceeded the logger's process controller stops restarting the logger, and instead performs an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the logger's process controller measures logger restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the logger's process controller stops restarting the logger. Instead the logger's process controller performs an action that is based on the value of the maxRestartDelay property.	120
maxRestartDelay	Determines the behavior of the logger's process controller when the rate of logger restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the logger's process controller is stopped. If you specify a value greater than zero, this is the number of seconds to wait before the restart history information held by the logger's process controller is reset and the logger is restarted.	-1
wmqfte.oracle.port	The port that the logger uses to connect to the Oracle instance. This port is also known as a TNS listener.	1521
wmqfte.oracle.host	The host that the logger uses to connect to the Oracle instance.	localhost
armELEMTYPE	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMTYPE parameter value specified in the associated ARM policy. For a logger, set ELEMTYPE to SYSBFGLG.	Not set
armELEMENT	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMENT parameter value specified in the associated ARM policy. You can set the ELEMENT value to correspond to the logger name.	Not set

Table 47. Connection properties for bindings mode (continued)

Property name	Description	Default value
loggerQMgrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the logger's coordination queue manager.	 For details on creating the Authentication Credentials File see "Configuring MQMFTCredentials.xml on z/OS" on page 42  See Configuring MQMFTCredentials.xml for information on the location and permissions of this file.  Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication
trace	Optional property. Trace specification when the logger is to be run with trace enabled at logger start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, com.ibm.wmqfte.databaselogger, com.ibm.wmqfte.databaselogger.operation=all. You can specify multiple trace specifications in a colon-separated list. For example, com.ibm.wmqfte.databaselogger=moderate:com.ibm.wmqfte.databaselogger.operation=all	None
traceFiles	Optional property. The total number of trace files to keep. This value applies to the process controller of a logger, as well as the logger itself.	5
traceSize	Optional property. The maximum size in MB of each trace file, before trace wraps onto the next file. This value applies to the process controller of the logger, and the logger itself.	20

Client mode connection properties

V 9.0.4

The following table shows the additional logger properties required to support client mode connection to a logger queue manager.

Table 48. Connection properties for client mode

Property name	Description	Default value
wmqfte.queue.manager.host	Host name, or IP address, of logger queue manager.	No default value
wmqfte.queue.manager.port	Port on which the logger queue manager is listening.	1414
wmqfte.queue.manager.channel	Name of the server connection channel on the logger queue manager.	SYSTEM.DEF.SVRCONN
wmqfte.Ssl.CipherSuite	Specifies TLS aspects of how the logger and the logger queue manager exchange data. The value of wmqfte.Ssl.CipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the logger queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.	No default value

Table 48. Connection properties for client mode (continued)

Property name	Description	Default value
wmqfte.Ssl.PeerName	Specifies a distinguished name skeleton that must match the name that is provided by the logger queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	No default value
wmqfte.Ssl.TrustStore	Specifies the location of the certificates that the logger trusts. The value of wmqfte.Ssl.TrustStore is a file path. If the file path is a Windows file path the backslash character (\) must be escaped with a further backslash character (\\). Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.TrustStoreCredentialsFile	The path to the file that contains the wmqfte.Ssl.TrustStore credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.TrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.KeyStore	Specifies the location of the private key of the logger. The value of wmqfte.Ssl.KeyStore is a file path. If the file path is a Windows file path the backslash character (\) must be escaped with a further backslash character (\\). Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStore.CredentialsFile	The path to the file that contains the wmqfte.Ssl.KeyStore credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.FipsRequired	Specifies that you want to enable FIPS support at the level of the logger. The value of this property can be true or false. For more information, see “FIPS support in MFT” on page 641 .	false

Related reference

[“The use of environment variables in MFT properties” on page 511](#)

From IBM WebSphere MQ 7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

Java system properties for MFT

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

Define system properties and other JVM options for the JVM that is to run Managed File Transfer commands by defining the environment variable `BFG_JVM_PROPERTIES`. For example, to set the `com.ibm.wmqfte.maxConsoleLineLength` property on a UNIX-type platform, define the variable as follows:

```
export BFG_JVM_PROPERTIES="-Dcom.ibm.wmqfte.maxConsoleLineLength=132"
```

If you are running an agent as a Windows service, you can modify the agent's Java system properties by specifying the `-sj` parameter on the **fteModifyAgent** command.

Table 49. Java system properties

Property name	Description	Value
<code>com.ibm.wmqfte.maxConsoleLineLength</code>	Maximum length of line that can be written to the console. Lines that exceed this length are word wrapped. This value is expressed in bytes (not characters).	 The default length for IBM i is 132 bytes.   For UNIX, Linux, Windows, and z/OS, the length is unlimited.
<code>com.ibm.wmqfte.daemon.windows.windowsServiceLogFilesm</code>	(Windows only.) Specifies the maximum number of Windows service log files to keep. Windows service log files are created in the agent and database logger logs directories if these applications are running as a Windows service. Windows service log files are named with the prefix <i>service</i> , and contain messages about the starting and stopping of the service.	5

Related concepts

[“MFT configuration options on Multiplatforms” on page 56](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

[“Hints and tips for using MFT” on page 261](#)

Here are some suggestions to help you to make best use of Managed File Transfer.

SSL properties for MFT

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

The following MFT properties files include SSL properties:

- [The `agent.properties` file](#)
- [The `coordination.properties` file](#)
- [The `command.properties` file](#)

- **V 9.0.4** The `logger.properties` file

For information about using SSL with Managed File Transfer, see [“Configuring SSL or TLS encryption for MFT”](#) on page 51.

For IBM WebSphere MQ 7.5 or later, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product to vary depending on environment changes, such as which user is running the process. For more information, see [“The use of environment variables in MFT properties”](#) on page 511.

SSL properties for the `agent.properties` file

The `agent.properties` file for an agent is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name` directory. The properties that it contains include the following SSL properties:

Property name	Description	Default value
<code>agentSslCipherSpec</code>	<p>Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the agent and the agent queue manager.</p> <p>The value of <code>agentSslCipherSpec</code> is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the agent queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS.</p> <p><code>agentSslCipherSpec</code> is similar to <code>agentSslCipherSuite</code>. If both <code>agentSslCipherSuite</code> and <code>agentSslCipherSpec</code> are specified, the value of <code>agentSslCipherSpec</code> is used.</p>	None
<code>agentSslCipherSuite</code>	<p>Specifies SSL aspects of how the agent and the agent queue manager exchange data.</p> <p>The value of <code>agentSslCipherSuite</code> is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.</p> <p><code>agentSslCipherSuite</code> is similar to <code>agentSslCipherSpec</code>. If both <code>agentSslCipherSuite</code> and <code>agentSslCipherSpec</code> are specified, the value of <code>agentSslCipherSpec</code> is used.</p>	None
<code>agentSslPeerName</code>	<p>Specifies a distinguished name skeleton that must match the name that is provided by the agent queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.</p>	None
<code>agentSslTrustStore</code>	<p>Specifies the location of the certificates that the agent trusts. The value of <code>agentSslTrustStore</code> is a file path. If it is a Windows file path the backslash character (\) must be escaped (\\).</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	None

Table 50. SSL properties for the agent.properties file (continued)

Property name	Description	Default value
agentSslKeyStore	Specifies the location of the private key of the agent. The value of agentSslKeyStore is a file path. If it is a Windows file path the backslash character (\) must be escaped (\\). This property is only required if the agent queue manager requires client authentication. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	None
agentSslFipsRequired	Specifies that you want to enable FIPS support at the level of the agent. The value of this property can be true or false. For more information, see "FIPS support in MFT" on page 641.	false
agentSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
agentSslKeyStoreCredentialsFile	The path to the file that contains the agentSslKeyStore credential. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	The default value for this property is %USERPROFILE%\MQMFTCredentials.xml on Windows and \$HOME/MQMFTCredentials.xml on other platforms.
agentSslTrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
agentSslTrustStoreCredentialsFile	The path to the file that contains the agentSslTrustStore credential. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	The default value for this property is %USERPROFILE%\MQMFTCredentials.xml on Windows and \$HOME/MQMFTCredentials.xml on other platforms.

SSL properties for the coordination.properties file

The `coordination.properties` file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory. The properties that it contains include the following SSL properties:

Table 51. SSL properties for the coordination.properties file

Property name	Description	Default value
coordinationSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the coordination queue manager. The value of coordinationSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the coordination queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS . coordinationSslCipherSpec is similar to coordinationSslCipherSuite. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.	None

Table 51. SSL properties for the coordination.properties file (continued)

Property name	Description	Default value
coordinationSslCipherSuite	<p>Specifies SSL aspects of how the commands and the coordination queue manager exchange data.</p> <p>The value of coordinationSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.</p> <p>coordinationSslCipherSuite is similar to coordinationSslCipherSpec. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.</p>	None
coordinationSslPeerName	<p>Specifies a distinguished name skeleton that must match the name that is provided by the coordination queue manager. The distinguished name is used to check the identifying certificate that is presented by the coordination queue manager on connection.</p>	None
coordinationSslTrustStore	<p>Specifies the location of the certificates that the commands trust. The value of coordinationSslTrustStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\).</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	None
coordinationSslTrustStoreType	<p>The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.</p>	jks
coordinationSslTrustStoreCredentialsFile	<p>The path to the file that contains the coordinationSslTrustStore credentials.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	The default value for this property is %USERPROFILE%\MQMFTCredentials.xml on Windows and \$HOME/MQMFTCredentials.xml on other platforms.
coordinationSslKeyStore	<p>Specifies the location of the private key of the commands. The value of coordinationSslKeyStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). This property is only required if the coordination queue manager requires client authentication.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	None
coordinationSslKeyStoreType	<p>The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.</p>	jks
coordinationSslKeyStoreCredentialsFile	<p>The path to the file that contains the coordinationSslKeyStore credentials.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	The default value for this property is %USERPROFILE%\MQMFTCredentials.xml on Windows and \$HOME/MQMFTCredentials.xml on other platforms.
coordinationSslFipsRequired	<p>Specifies that you want to enable FIPS support at the level of the coordination queue manager. The value of this property can be true or false. For more information, see “FIPS support in MFT” on page 641.</p>	false

SSL properties for the `command.properties` file

The `command.properties` file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory. The properties that it contains include the following SSL properties:

Property name	Description	Default value
<code>connectionSslCipherSpec</code>	<p>Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the command queue manager.</p> <p>The value of <code>connectionSslCipherSpec</code> is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the command queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS.</p> <p><code>connectionSslCipherSpec</code> is similar to <code>connectionSslCipherSuite</code>. If both <code>connectionSslCipherSuite</code> and <code>connectionSslCipherSpec</code> are specified, the value of <code>connectionSslCipherSpec</code> is used.</p>	None
<code>connectionSslCipherSuite</code>	<p>Specifies SSL aspects of how the commands and the command queue manager exchange data.</p> <p>The value of <code>connectionSslCipherSuite</code> is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.</p> <p><code>connectionSslCipherSuite</code> is similar to <code>connectionSslCipherSpec</code>. If both <code>connectionSslCipherSuite</code> and <code>connectionSslCipherSpec</code> are specified, the value of <code>connectionSslCipherSpec</code> is used.</p>	None
<code>connectionSslPeerName</code>	<p>Specifies a distinguished name skeleton that must match the name that is provided by the command queue manager. The distinguished name is used to check the identifying certificate that is presented by the command queue manager on connection.</p>	None
<code>connectionSslTrustStore</code>	<p>Specifies the location of the certificates that the commands trust. The value of <code>connectionSslTrustStore</code> is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\).</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	None
<code>connectionSslTrustStoreType</code>	<p>The type of SSL truststore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either <code>jks</code> or <code>pkcs12</code>.</p>	<code>jks</code>
<code>connectionSslTrustStoreCredentialsFile</code>	<p>The path to the file that contains the <code>connectionSslTrustStore</code> credentials.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	The default value for this property is <code>%USERPROFILE%\MQMFTCredentials.xml</code> on Windows and <code>\$HOME/MQMFTCredentials.xml</code> on other platforms.
<code>connectionSslKeyStore</code>	<p>Specifies the location of the private key of the commands. The value of <code>connectionSslKeyStore</code> is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). This property is only required if the command queue manager requires client authentication.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>	None

Table 52. SSL properties for the <code>command.properties</code> file (continued)		
Property name	Description	Default value
<code>connectionSslKeyStoreType</code>	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either <code>jks</code> or <code>pkcs12</code> . For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	<code>jks</code>
<code>connectionSslKeyStoreCredentialsFile</code>	The path to the file that contains the <code>connectionSslKeyStore</code> credentials. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	The default value for this property is <code>%USERPROFILE%\MQMFTcredentials.xml</code> on Windows and <code>\$HOME/MQMFTcredentials.xml</code> on other platforms.
<code>connectionSslFipsRequired</code>	Specifies that you want to enable FIPS support at the level of the command queue manager. The value of this property can be <code>true</code> or <code>false</code> . For more information, see “FIPS support in MFT” on page 641 .	<code>false</code>

SSL properties for the `logger.properties` file

V 9.0.4

The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. The properties required to support client mode connection to a logger queue manager include the following SSL properties:

Table 53. SSL properties for the <code>logger.properties</code> file		
Property name	Description	Default value
<code>wmqfte.Ssl.CipherSuite</code>	Specifies TLS aspects of how the logger and the logger queue manager exchange data. The value of <code>wmqfte.Ssl.CipherSuite</code> is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the logger queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.	No default value
<code>wmqfte.Ssl.PeerName</code>	Specifies a distinguished name skeleton that must match the name that is provided by the logger queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	No default value
<code>wmqfte.Ssl.TrustStore</code>	Specifies the location of the certificates that the logger trusts. The value of <code>wmqfte.Ssl.TrustStore</code> is a file path. If the file path is a Windows file path the backslash character (<code>\</code>) must be escaped with a further backslash character (<code>\\</code>). Note that the value of this property can contain environment variables.	No default value
<code>wmqfte.Ssl.TrustStoreCredentialsFile</code>	The path to the file that contains the <code>wmqfte.Ssl.TrustStore</code> credential. Note that the value of this property can contain environment variables.	No default value

Table 53. SSL properties for the `logger.properties` file (continued)

Property name	Description	Default value
<code>wmqfte.Ssl.TrustStoreType</code>	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either <code>jks</code> or <code>pkcs12</code> .	<code>jks</code>
<code>wmqfte.Ssl.KeyStore</code>	Specifies the location of the private key of the logger. The value of <code>wmqfte.Ssl.KeyStore</code> is a file path. If the file path is a Windows file path the backslash character (<code>\</code>) must be escaped with a further backslash character (<code>\\</code>). Note that the value of this property can contain environment variables.	No default value
<code>wmqfte.Ssl.KeyStore.CredentialsFile</code>	The path to the file that contains the <code>wmqfte.Ssl.KeyStore</code> credential. Note that the value of this property can contain environment variables.	No default value
<code>wmqfte.Ssl.KeyStoreType</code>	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either <code>jks</code> or <code>pkcs12</code> .	<code>jks</code>
<code>wmqfte.Ssl.FipsRequired</code>	Specifies that you want to enable FIPS support at the level of the logger. The value of this property can be <code>true</code> or <code>false</code> . For more information, see “FIPS support in MFT” on page 641 .	<code>false</code>

SHA-2 CipherSpecs and CipherSuites for MFT

Managed File Transfer supports SHA-2 CipherSpecs and CipherSuites.

For more information about CipherSpecs and CipherSuites that are available for connections between agents and IBM MQ queue managers, see [SSL CipherSpecs and CipherSuites](#).

For more information about configuring CipherSpecs and CipherSuites for use with the protocol bridge agent and FTPS servers, see [“FTPS server support by the protocol bridge” on page 639](#) and [“Protocol bridge properties file format” on page 553](#).

If you want to comply with SP 800-131A, you must satisfy the following requirements:

- You must use FTPS, which you have configured appropriately; SFTP is not supported.
- The remote server must send SP 800-131A-compliant cipher suites only.

Related reference

[“SSL properties for MFT” on page 580](#)

Use SSL or TLS with IBM MQ and Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

MFT file logger configuration files

In addition to the `logger.properties` file, a Managed File Transfer stand-alone file logger also has an XML configuration file in its configuration directory. This configuration file is called `FileLoggerFormat.xml` and it defines the format used by the file logger to write messages to the log file. The content of this file must conform to the XML schema defined in the `FileLoggerFormat.xsd` file.

MFT stand-alone file logger default log format

Default log file format definition for the Managed File Transfer stand-alone file logger.

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00" xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <callCompleted>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="3" ignoreNull="false">/transaction/status/@resultCode</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/job/name</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/command/
@type</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/command/
@name</insert>
          <insert type="system" width="0" ignoreNull="true">callArguments</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/callResult/
@outcome</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/callResult/
result/error</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callCompleted>
    <callStarted>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user" width="0" ignoreNull="false">/transaction/job/name</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/command/
@type</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/transferSet/call/command/
@name</insert>
          <insert type="system" width="0" ignoreNull="true">callArguments</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callStarted>
    <monitorAction>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user" width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="3" ignoreNull="false">/monitorLog/status/@resultCode</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/action</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </monitorAction>
    <monitorCreate>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user" width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user" width="0" ignoreNull="false">/monitorLog/action</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </monitorCreate>
    <monitorFired>
      <format>
        <inserts>
```



```

        <insert type="user" width="0" ignoreNull="false">/schedulelog/originator/userID</insert>
        <insert type="user" width="0" ignoreNull="true">/schedulelog/schedule/submit</insert>
        <insert type="user" width="0" ignoreNull="true">/schedulelog/schedule/submit/@timezone</
insert>
        <insert type="user" width="3" ignoreNull="true">/schedulelog/schedule/repeat/frequency</
insert>
        <insert type="user" width="12" ignoreNull="true">/schedulelog/schedule/repeat/frequency/
@interval</insert>
        <insert type="user" width="3" ignoreNull="true">/schedulelog/schedule/repeat/expireCount</
insert>
        <insert type="user" width="0" ignoreNull="true">/schedulelog/status/supplement</insert>
    </inserts>
    <separator>;</separator>
</format>
</scheduleSubmitInfo>
<scheduleSubmitTransfer>
    <format>
        <inserts>
            <insert type="user" width="19" ignoreNull="false">/schedulelog/action/@time</insert>
            <insert type="user" width="48" ignoreNull="false">/schedulelog/@ID</insert>
            <insert type="system" width="10" ignoreNull="false">type</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/sourceAgent/@agent |
/transaction/sourceWebUser/@webGatewayAgentName |
/transaction/sourceWebGateway/@webGatewayAgentName</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/sourceAgent/@QMgr |
/transaction/sourceWebUser/@webGatewayAgentQMgr |
/transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/destinationAgent/@agent |
/transaction/destinationWebUser/@webGatewayAgentName |
/transaction/destinationWebGateway/@webGatewayAgentName</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/destinationAgent/@QMgr |
/transaction/destinationWebUser/@webGatewayAgentQMgr |
/transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
        </inserts>
        <separator>;</separator>
    </format>
</scheduleSubmitTransfer>
<scheduleSubmitTransferSet>
    <format>
        <inserts>
            <insert type="user" width="19" ignoreNull="false">/schedulelog/action/@time</insert>
            <insert type="user" width="48" ignoreNull="false">/schedulelog/@ID</insert>
            <insert type="system" width="10" ignoreNull="false">type</insert>
            <insert type="user" width="0" ignoreNull="false">source/file | source/queue</insert>
            <insert type="user" width="5" ignoreNull="true">source/@type</insert>
            <insert type="user" width="6" ignoreNull="true">source/@disposition</insert>
            <insert type="user" width="0" ignoreNull="false">destination/file | destination/queue</
insert>
            <insert type="user" width="5" ignoreNull="true">destination/@type</insert>
            <insert type="user" width="9" ignoreNull="true">destination/@exist</insert>
        </inserts>
        <separator>;</separator>
    </format>
</scheduleSubmitTransferSet>
<transferStarted>
    <format>
        <inserts>
            <insert type="user" width="19" ignoreNull="false">/transaction/action/@time</insert>
            <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
            <insert type="system" width="6" ignoreNull="false">type</insert>
            <insert type="user" width="3" ignoreNull="true">/transaction/status/@resultCode</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/sourceAgent/@agent |
/transaction/sourceWebUser/@webGatewayAgentName |
/transaction/sourceWebGateway/@webGatewayAgentName</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/sourceAgent/@QMgr |
/transaction/sourceWebUser/@webGatewayAgentQMgr |
/transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/sourceAgent/@agentType |
/transaction/sourceWebUser/@webGatewayAgentType |
/transaction/sourceWebGateway/@webGatewayAgentType</insert>
            <insert type="user" width="0" ignoreNull="false">/transaction/destinationAgent/@agent |
/transaction/destinationWebUser/@webGatewayAgentName |
/transaction/destinationWebGateway/@webGatewayAgentName</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/destinationAgent/@QMgr |
/transaction/destinationWebUser/@webGatewayAgentQMgr |
/transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/originator/userID</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/job/name</insert>
            <insert type="user" width="0" ignoreNull="true">/transaction/scheduleLog/@ID</insert>
        </inserts>
        <separator>;</separator>
    </format>

```



```

    /transaction/destinationWebUser/@webGatewayAgentName |
    /transaction/destinationWebGateway/@webGatewayAgentName</insert>
    <insert type="user" width="0" ignoreNull="true">/transaction/destinationAgent/@QMgr |
    /transaction/destinationWebUser/@webGatewayAgentQMgr |
    /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
    <insert type="user" width="0" ignoreNull="true">/transaction/destinationAgent/@agentType |
    /transaction/destinationWebUser/@webGatewayAgentType |
    /transaction/destinationWebGateway/@webGatewayAgentType</insert>
    <insert type="user" width="0" ignoreNull="true">/transaction/originator/userID</insert>
    <insert type="user" width="0" ignoreNull="true">/transaction/job/name</insert>
    <insert type="user" width="0" ignoreNull="true">/transaction/status/supplement</insert>
  </inserts>
  <separator>;</separator>
</format>
</transferDelete>
<transferProgress>
  <format>
    <inserts>
      <insert type="user" width="19" ignoreNull="false">/transaction/action/@time</insert>
      <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system" width="6" ignoreNull="false">type</insert>
      <insert type="user" width="3" ignoreNull="true">status/@resultCode</insert>
      <insert type="user" width="0" ignoreNull="false">source/file | source/queue</insert>
      <insert type="user" width="0" ignoreNull="false">source/file/@size | source/queue/@size</
insert>
      <insert type="user" width="5" ignoreNull="true">source/@type</insert>
      <insert type="user" width="6" ignoreNull="true">source/@disposition</insert>
      <insert type="user" width="0" ignoreNull="true">source/file/@alias | source/queue/@alias</
insert>
      <insert type="user" width="0" ignoreNull="true">source/file/@filesystem | source/queue/
@filesystem</insert>
      <insert type="user" width="0" ignoreNull="true">source/@correlationBoolean1</insert>
      <insert type="user" width="0" ignoreNull="true">source/@correlationNum1</insert>
      <insert type="user" width="0" ignoreNull="true">source/@correlationString1</insert>
      <insert type="user" width="0" ignoreNull="false">destination/file | destination/queue</
insert>
      <insert type="user" width="0" ignoreNull="false">destination/file/@size | destination/queue/
@size</insert>
      <insert type="user" width="5" ignoreNull="true">destination/@type</insert>
      <insert type="user" width="9" ignoreNull="true">destination/@exist</insert>
      <insert type="user" width="0" ignoreNull="true">destination/file/@alias | destination/queue/
@alias</insert>
      <insert type="user" width="0" ignoreNull="true">destination/file/@filesystem | destination/
queue/@filesystem</insert>
      <insert type="user" width="0" ignoreNull="true">destination/file/@truncateRecords</insert>
      <insert type="user" width="0" ignoreNull="true">destination/@correlationBoolean1</insert>
      <insert type="user" width="0" ignoreNull="true">destination/@correlationNum1</insert>
      <insert type="user" width="0" ignoreNull="true">destination/@correlationString1</insert>
      <insert type="user" width="0" ignoreNull="true">status/supplement</insert>
    </inserts>
    <separator>;</separator>
  </format>
</transferProgress>
</messageTypes>
</logFormatDefinition>

```

Related reference

[“MFT stand-alone file logger format” on page 102](#)

The format of message information written by the file logger can be defined in the FileLoggerFormat.xml file.

[“Stand-alone file logger format XSD” on page 591](#)

The schema for a stand-alone file format.

Stand-alone file logger format XSD

The schema for a stand-alone file format.

Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

```

Copyright IBM Corp. 2011, 2024. All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

@end_non_restricted_prolog@

-->

<!--

This schema defines the format of the FileLoggerFormat XML file that contains the definition of the format to use when logging FTE log messages to a file. When an XML file that conforms to this schema is processed by a file logger it can contain definitions for one or more message type(s) that define how log messages of those types are output to the file log.

-->

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="fteutils.xsd"/>

<!--

Defines the logFileDefinition and version number
<logFileDefinition version="1.00" ...
 <messageTypes>

 ...

 </messageTypes>
</logFileDefinition>

-->

<xsd:element name="logFileDefinition">

 <xsd:complexType>

 <xsd:sequence>

 <xsd:element name="messageTypes" type="messageTypesType" maxOccurs="1" minOccurs="1"/>

 </xsd:sequence>

 <xsd:attribute name="version" type="versionType" use="required"/>

 </xsd:complexType>

</xsd:element>

<!--

Defines the set of accepted message types. The definition of individual message types is optional. If a particular types element is present but empty then no line will be output for messages of that type. If a particular types element is not present then the default format will be used to format messages of that type.

-->

<xsd:complexType name="messageTypesType">

 <xsd:sequence>

 <xsd:element name="callCompleted" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="callStarted" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="monitorAction" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="monitorCreate" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="monitorFired" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="notAuthorized" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleDelete" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleExpire" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleSkipped" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleSubmitInfo" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleSubmitTransfer" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="scheduleSubmitTransferSet" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="transferStarted" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="transferCancelled" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="transferComplete" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="transferDelete" type="messageType" maxOccurs="1" minOccurs="0"/>

 <xsd:element name="transferProgress" type="messageType" maxOccurs="1" minOccurs="0"/>

 </xsd:sequence>

</xsd:complexType>

```

<!--
  Defines the content of a message type definition e.g.

  <callStarted>
  <format>
  ...
  </format>
  <callStarted>
-->
<xsd:complexType name="messageType">
  <xsd:sequence>
    <xsd:element name="format" type="messageFormatType" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Defines the content of a message format definition e.g.

  <format>
  <inserts>
  ...
  </inserts>
  <separator>;</separator>
  </format>
-->
<xsd:complexType name="messageFormatType">
  <xsd:sequence>
    <xsd:element name="inserts" type="insertsType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="separator" type="scheduleType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Defines the content of the inserts element e.g.

  <inserts>
  <insert ...>
  <insert ...>
  ...
  </inserts>
-->
<xsd:complexType name="insertsType">
  <xsd:sequence>
    <xsd:element name="insert" type="insertType" maxOccurs="unbounded" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Defines the content of an insert definition e.g.

  <insert type="user" width="0" ignoreNull="true">/transaction/@ID</insert>
-->
<xsd:complexType name="insertType">
  <xsd:attribute name="type" type="insertTypeType" use="required"/>
  <xsd:attribute name="width" type="xsd:nonNegativeInteger" use="required"/>
  <xsd:attribute name="ignoreNull" type="xsd:boolean" use="required"/>
</xsd:complexType>

<!--
  Defines the accepted choices for the insert type attribute.
-->
<xsd:simpleType name="insertTypeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="user"/>
    <xsd:enumeration value="system"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Related reference

[“MFT stand-alone file logger format” on page 102](#)

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

[“MFT stand-alone file logger default log format” on page 587](#)

Default log file format definition for the Managed File Transfer stand-alone file logger.

SYSTEM.FTE topic

The SYSTEM.FTE topic is a topic on the coordination queue manager that Managed File Transfer uses to log transfers and store information about agents, monitors, schedules, and templates.

Topic structure

```
SYSTEM.FTE
  /Agents
    /agent_name
  /monitors
    /agent_name
  /Scheduler
    /agent_name
  /Templates
    /template_ID
  /Transfers
    /agent_name
    /transfer_ID
  /Log
    /agent_name
      /Monitors
      /schedule_ID
      /transfer_ID
```

SYSTEM.FTE/Agents/agent_name

This topic contains a retained publication that describes an agent in your Managed File Transfer network and its properties. The message on this topic is updated periodically with the agent status. For more information, see [“MFT agent status message format” on page 709](#).

SYSTEM.FTE/monitors/agent_name

This topic contains retained publications that describe the resource monitors associated with the agent *agent_name*. The XML of the retained publication conforms to the schema `MonitorList.xsd`. For more information, see [“MFT monitor list message format” on page 711](#).

SYSTEM.FTE/Scheduler/agent_name

This topic contains a retained publication that describes all of the active schedules that are associated with the agent *agent_name*. The XML of the retained publication conforms to the schema `ScheduleList.xsd`. For more information, see [“MFT schedule list message format” on page 715](#).

SYSTEM.FTE/Templates

This topic contains retained publications that describe all of the templates that are defined in your Managed File Transfer topology.

- The publication that is associated with each template is published to a subtopic with the name `SYSTEM.FTE/Templates/template_ID`.

For an example of the contents of this retained publication, see [“MFT example template XML message” on page 719](#).

SYSTEM.FTE/Transfers/agent_name

This topic contains publications that describe that status of transfers that originate at the agent *agent_name*. The publications that are associated with each transfer are published to a subtopic with the name `SYSTEM.FTE/Transfers/agent_name/transfer_ID`. These publications are used by the IBM MQ Explorer plug-in to provide progress information about individual transfers. The XML of the publication conforms to the schema `TransferStatus.xsd`. For more information, see [“File transfer status message format” on page 719](#).

SYSTEM.FTE/Log/agent_name

This topic contains publications that log information about transfers, monitors, and schedules that originate at the agent *agent_name*. These publications can be logged by the database logger to provide audit records of the events that happen in your Managed File Transfer network.

- The publications that are associated with each transfer are published to a subtopic with the name `SYSTEM.FTE/Log/agent_name/transfer_ID` and the XML of the publication conforms to the schema `TransferLog.xsd`. For more information, see [“File transfer log message formats”](#) on page 723.
- The publications that are associated with each scheduled transfer are published to a subtopic with the name `SYSTEM.FTE/Log/agent_name/schedule_ID` and the XML of the publication conforms to the schema `ScheduleLog.xsd`. For more information, see [“Scheduled file transfer log message formats”](#) on page 744.
- The publications that are associated with each monitor are published to a subtopic with the name `SYSTEM.FTE/Log/agent_name/Monitors/monitor_name/monitor_ID` and the XML of the publication conforms to the schema `MonitorLog.xsd`. For more information, see [“MFT monitor log message format”](#) on page 749.

MFT Agent queue settings

The MQSC command scripts generated by the `fteCreateAgent` command create the agent queues with parameters set to the following values. If you do not use the MQSC scripts provided to create the queues, but create the queues manually, ensure you set the following parameters to the values given.

Agent operation queues

The agent's operation queues have the following names:

- `SYSTEM.FTE.COMMAND.agent_name`
- `SYSTEM.FTE.DATA.agent_name`
- `SYSTEM.FTE.EVENT.agent_name`
- `SYSTEM.FTE.REPLY.agent_name`
- `SYSTEM.FTE.STATE.agent_name`

Parameter	Value (if applicable)
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	5000
MAXMSGL	4194304
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Agent authority queues

The agent's authority queues have the following names:

- `SYSTEM.FTE.AUTHADM1.agent_name`

- SYSTEM.FTE.AUTHAGT1.agent_name
- SYSTEM.FTE.AUTHMON1.agent_name
- SYSTEM.FTE.AUTHOPS1.agent_name
- SYSTEM.FTE.AUTHSCH1.agent_name
- SYSTEM.FTE.AUTHTRN1.agent_name

<i>Table 55. Agent authority queue parameters</i>	
Parameter	Value (if applicable)
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	0
MAXMSGL	0
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Related reference

“[fteCreateAgent \(create an MFT agent\)](#)” on page 371

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

MFT system queues and the system topic

Managed File Transfer has a number of system queues and one system topic that are for internal use only.

Any queues with a name beginning SYSTEM.FTE are internal system queues for Managed File Transfer (MFT). Do not delete these queues, as doing so prevents IBM MQ MFT from working correctly. [Table 56 on page 596](#) shows which type of message is on each queue:

<i>Table 56. Queue names, type and usage</i>		
Queue name	Queue type	Usage
SYSTEM.FTE.AUTHAGT1.agent_name	Authority	Queue for configuring authority for sending and receiving transfer requests.
SYSTEM.FTE.AUTHTRN1.agent_name	Authority	Queue for configuring authority to start and cancel managed transfers. Also to start managed calls.
SYSTEM.FTE.AUTHMON1.agent_name	Authority	Queue for configuring authority to allow a user to create or delete resource monitors that were created by the same user.

Table 56. Queue names, type and usage (continued)

Queue name	Queue type	Usage
SYSTEM.FTE.AUTHOPS1.agent_name	Authority	Queue for configuring authority to delete resource monitors and scheduled transfers that were created by another user.
SYSTEM.FTE.AUTHSCH1.agent_name	Authority	Queue for configuring authority to create or delete scheduled transfers that were created by the same user.
SYSTEM.FTE.AUTHADM1.agent_name	Authority	Queue for configuring authority to shut down the agent, using the -m option on the fteStopAgent command.
SYSTEM.FTE.COMMAND.agent_name	Operation	Queue for sending command requests to an agent.
SYSTEM.FTE.DATA.agent_name	Operation	Queue used by a destination agent for holding data sent by a source agent.
SYSTEM.FTE.REPLY.agent_name	Operation	Queue for receiving replies from a destination agent.
SYSTEM.FTE.STATE.agent_name	Operation	Queue for holding the status of a transfer request.
SYSTEM.FTE.EVENT.agent_name	Operation	Queue for holding resource monitor history.

If an agent is participating in message-to-file or file-to-message transfers, the definition of the SYSTEM.FTE.STATE.agent_name queue might need to be modified to allow these managed transfers to take place. For more information on this, see [Guidance for setting MQ attributes and MFT properties associated with message size](#).



Attention: You should not change the definitions of the other system queues.

Also, do not modify or delete the SYSTEM.FTE topic as this is also for internal use only.

Temporary queues

Managed File Transfer creates temporary queues for a number of purposes. The name of each queue starts with WMQFTE. by default. (The period is part of the default prefix.) If you want to change this prefix, you can use the **dynamicQueuePrefix** property in the command.properties file or the coordination.properties file or both. The property in the command.properties file is used to set the prefix of temporary queues that are created for responses to commands that require a response from the agent. The property in the coordination.properties file is used to set the prefix of temporary queues that are created for other purposes; for example, the WMQFTE.FTE.TIMECHCK.QUEUE, where WMQFTE. is the value defined by the **dynamicQueuePrefix** property.

Related reference

[“Restricting user authorities on MFT agent actions” on page 342](#)

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

MFT object naming conventions

Use the following naming conventions for your Managed File Transfer objects:

- Agent and logger names:
 - Can be a maximum of 28 characters long and are not case-sensitive.
 - Entered in lowercase or mixed case are converted to uppercase
 - Must conform to standard IBM MQ object naming conventions.

These conventions are detailed as follows: [Rules for naming IBM MQ objects](#).
- In addition to the IBM MQ object naming conventions, the:
 - Forward slash (/) character cannot be used in agent names or logger names
 - Percent (%) character cannot be used in agent names or logger names.
- The names of properties in the properties files are case-sensitive.
- Queue manager names are case-sensitive.
- File names are case-sensitive for some platforms.
- Resource monitor and transfer template names:
 - Are not case-sensitive
 - Entered in lowercase or mixed case are converted to uppercase
 - Must not contain asterisk (*), percent (%) or question mark (?) characters
- Protocol file server names must:
 - Be a minimum of 2 characters long, but there is no maximum length limit
 - Are not case-sensitive
 - Must conform to standard IBM MQ object naming conventions.

These conventions are detailed as follows: [Rules for naming IBM MQ objects](#).

Files in the IBM i integrated file system (IFS)

 File names in the IFS cannot contain any of the following characters:

- Backslash (\)
- Forward slash (/)
- Colon (:)
- Asterisk (*)
- Question mark (?)
- Quotation marks (")
- Less than symbol (<)
- Greater than symbol (>)
- Vertical bar (|)

If you attempt to transfer files with names containing any of these characters to an IBM i IFS, the transfer of these files fails.

Data set names

z/OS Data sets have naming restrictions, which affect the maximum name length and the available characters that you can use for data set names. PDS data set member names can be a maximum of eight characters and cannot contain the dot (.) character. When you transfer to a data set, you must explicitly specify the name, which means these naming restrictions do not cause a problem. But when you transfer from files to PDS members the file path might not map to a PDS member name. When you transfer to a PDS data set, each source file becomes a PDS member and each member name is generated from the name of the source.

PDS member names are z/OS unqualified names and are defined by the following regular expression:

```
[a-zA-Z$#@] [a-zA-Z0-9$#@]{0-7}
```

The following scheme is used to convert a source data set or source file name to a valid PDS member name. The considerations are applied in the order listed:

1. Only the characters in the name after the last forward slash (/), the last backslash (\), or the last colon (:) character, are used. That is, only the name part of a file path is used.
2. For source files (not data sets or PDS members), the characters after and including the last dot (.) character, are ignored.
3. For any name longer than eight characters, the last eight characters only are used.
4. Dot characters are replaced with at sign (@) characters.
5. Invalid characters are replaced with at sign (@) characters.
6. If the conversion produces no characters, the PDS member name is @.

MFT administration reference

MFT agent status values

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

ACTIVE

The agent is running and is sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

READY

The agent is running, but is not sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

STARTING

The agent is starting, but is not yet ready to perform transfers.

UNREACHABLE

Agent status updates were not received at the expected time intervals. The agent might have stopped running due to an error, have been shut down abruptly, or might be running but is experiencing communication problems. For more information, see [“What to do if an agent is shown as being in an UNKNOWN state” on page 290.](#)

STOPPED

The agent has been stopped. It was shut down in a controlled manner.

ENDED UNEXPECTEDLY

The agent has ended unexpectedly. The agent will be automatically restarted, unless there have been more than `maxRestartCount` restarts within the `maxRestartInterval` time period and the `maxRestartDelay` value is less than or equal to 0. For more information about these properties, see [“The MFT agent.properties file” on page 526.](#)

NO_INFORMATION

The agent version might be IBM WebSphere MQ File Transfer Edition 7.0.2 or earlier. The agent is not publishing updates in a form that this command can process.

UNKNOWN

The status of the agent cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

V 9.0.2 At IBM MQ 9.0.2, when you run commands or look at the list of agents connecting to a coordination manager and their individual properties, you can see a new **Status Age** value for the agent that shows the age of their last reported status. For more information, see [“What to do if an agent is shown as being in an UNKNOWN state”](#) on page 290.

PROBLEM

The agent command handler might not be working. The agent is publishing status messages, but these status messages are out of date.

Related reference

[“fteListAgents: list the MFT agents for a coordination queue manager”](#) on page 452

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“fteShowAgentDetails”](#) on page 495

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

[“What to do if you think that your file transfer is stuck”](#) on page 299

On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

[“What to do if an agent is shown as being in an UNKNOWN state”](#) on page 290

Your agent is running and responds successfully to the **ftePingAgent** command, and items are being transferred normally. However, the **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer Managed File Transfer plugin, report the agent as being in an UNKNOWN state.

Multi MFT process controller overview

The IBM MQ Managed File Transfer (MFT) process controller is responsible for starting an MFT agent, and restarting that process if it ends for any reason. There is one process controller for every agent process.

Note: The process controller is applicable to IBM MQ for Multiplatforms only.

z/OS On IBM MQ for z/OS the agent process is restarted by Automatic Restart Manager (ARM). For more information on this, see [Configuring MFT for the z/OS Automatic Restart Manager \(ARM\)](#)

How the process controller works

When the **fteStartAgent** command is run, it starts up an instance of the process controller for that agent and the process controller then starts the agent process.

When the **fteStopAgent** command is run, it connects to the process controller for that agent and sends it a stop request. The process controller receives the request, stops the agent process and then shuts itself down.

The process controller monitors the agent process. If the agent process stops unexpectedly, the process controller restarts it.

By default, if an agent process stops five times within a two minute period, the process controller shuts itself down and does not try to restart the agent again. In this situation, you need to restart the agent manually, using the **fteStartAgent** command.

You can change this behavior by modifying the following agent properties:

- **maxRestartCount**
- **maxRestartDelay**
- **maxRestartInterval**

If you have configured an agent to connect to its agent queue manager using the BINDINGS transport, the process controller creates a connection to this queue manager when it starts up. The process controller then monitors this connection.

If the connection is broken because the queue manager has become unavailable, the process controller stops the agent and then attempts to re-establish the connection at regular intervals.

The time period between reconnection attempts is determined by the agent property **agentQMgrRetryInterval**. Once the queue manager is available again and the process controller has been able to connect to it, the process controller restarts the agent process.

Note: When an agent is configured to connect to its agent queue manager using the CLIENT transport, the agent process remains active if it becomes disconnected from the queue manager. In this situation, the agent process tries to reconnect itself at regular intervals.

For more information on the four properties mentioned in this section, see the [Advanced agent properties: Process controller](#) section of *The MFT agent . properties file* topic.

Process controller log files

The process controller writes informational messages to its event log. This is a file called pceventN.log, where N is a number, which can be found in the following directory: MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/

The size of each process controller event log file, and the number of historical files, is determined by the agent properties **outputLogSize** and **outputLogFiles**.

For more information on the properties mentioned in this section, see the [Advanced agent properties: Tracing and logging](#) section of *The MFT agent . properties file* topic.

Note: These properties are also used to determine the size and number of agent log files (called outputN.log) as well as the process controller log files.

The messages written to the process controller event log include the process identifier of the process controller, and the process identifier of the agent process. Some examples of these messages are shown here:

```
[21/06/2022 16:17:40.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0003I: IBM MQ Managed File Transfer process controller started.
Log files located at: C:\ProgramData\IBM\MQ\mqft\logs\QM1\agents\AGENT1.

[21/06/2022 16:17:55.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0007I: IBM MQ Managed File Transfer process controller with process
identifier 18736 started AGENT1@QM1 with process identifier 1748.

[21/06/2022 16:19:20.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0027W: Process has ended with return code 1 and will be
restarted to attempt to recover the problem.

[21/06/2022 16:19:20.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0007I: IBM MQ Managed File Transfer process controller with process
identifier 18736 started AGENT1@QM1 with process identifier 1304.
```

Here, the process controller associated with the agent AGENT1 was running with process identifier 18736.

Initially, it started the agent process – the process identifier for this process was 1748.

Shortly after the agent started, the process controller detected that it had stopped unexpectedly and so restarted it. Following the restart, the process identifier for the agent process is 1304.

Related reference

[“MFT process controller exit codes” on page 604](#)

If the Managed File Transfer process controller ends, a BFGPC0004I message is generated with an exit code that gives the reason why the process controller ended.

MFT agent process controller status values

The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

WAITING

The agent process controller is waiting for the queue manager to become available before starting the agent.

STARTED

The agent process controller has started the agent process.

STOPPED

The agent process controller has been stopped, either because of a request to stop the agent or because there have been too many agent process restarts within the restart interval.

RECOVERING

The agent process unexpectedly stopped and the process controller will attempt to restart it.

ISTOPPING

The agent process has received a request to shut down immediately. When the agent process has stopped, the process controller will stop.

CSTOPPING

The agent process has received a request to shut down in a controlled manner. When the agent process has stopped, the process controller will stop.

UNKNOWN

The agent process controller status cannot be determined. It might be that the agent process controller is not running, or that it is running on a different system from where the **fteShowAgentDetails** command was run.

Related reference

[“fteShowAgentDetails” on page 495](#)

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

MFT logger status values

The **fteShowLoggerDetails** commands produce logger status information. There are several possible values for this status.

ACTIVE

The logger is running and is sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

READY

The logger is running, but is not sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

STARTING

The logger is starting, but is not yet ready to perform transfers.

UNREACHABLE

Logger status updates were not received at the expected time intervals. The logger might have stopped running due to an error, or have been shut down abruptly, or be running but experiencing communication problems.

STOPPED

The logger has been stopped. It was shut down in a controlled manner.

ENDED UNEXPECTEDLY

The logger has ended unexpectedly. The logger will be automatically restarted, unless there have been more than `maxRestartCount` restarts within the `maxRestartInterval` time period and the `maxRestartDelay` value is less than or equal to 0. For more information about these properties, see [“MFT logger configuration properties” on page 111](#).

For the **`fteShowLoggerDetails`** command the details for the this status will include a status code, which is the logger process exit code. See "Process Exit Codes" for a list of known exit codes.

NO_INFORMATION

The logger version might be IBM WebSphere MQ File Transfer Edition 7.0.2 earlier. The logger is not publishing updates in a form that this command can process.

UNKNOWN

The status of the logger cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

PROBLEM

The logger command handler might not be working. The logger is publishing status messages, but these status messages are out of date.

Related reference

[“fteShowLoggerDetails” on page 502](#)

Use the **`fteShowLoggerDetails`** command to display the details of a particular Managed File Transfer logger.

MFT logger process controller status values

The **`fteShowLoggerDetails`** command produces logger process controller status information. There are several possible values for this status.

WAITING

The logger process controller is waiting for the queue manager to become available before starting the logger.

STARTED

The logger process controller has started the logger process.

STOPPED

The logger process controller has been stopped, either because of a request to stop the logger or because there have been too many logger process restarts within the restart interval.

RECOVERING

The logger process unexpectedly stopped and the process controller will attempt to restart it.

ISTOPPING

The logger process has received a request to shut down immediately. When the logger process has stopped, the process controller will stop.

CSTOPPING

The logger process has received a request to shut down in a controlled manner. When the logger process has stopped, the process controller will stop.

UNKNOWN

The logger process controller status cannot be determined. It might be that the logger process controller is not running, or that it is running on a different system from where the `fteShowLoggerDetails` command was run.

Related reference

[“fteShowLoggerDetails” on page 502](#)

Use the **fteShowLoggerDetails** command to display the details of a particular Managed File Transfer logger.

MFT process controller exit codes

If the Managed File Transfer process controller ends, a BFGPC0004I message is generated with an exit code that gives the reason why the process controller ended.

The following message appears to indicate that process controller has ended:

```
BFGPC0004I IBM MQ Managed File Transfer process controller ended with exit code reason_code.
```

where *reason_code* shows the reason why the process controller has ended.

Note: Exit codes from the process controller mostly reflect the standard operating system exit codes, but some exit codes are defined for specific purposes and are always accompanied with a specific message in the process controller log file.

Reason code	Description
RC_SUCCESS = 0	The process controller ended successfully.
RC_FAILURE = 1	General process controller failure return code (should in general not be returned).
RC_EXIT = 2	The process controller was forced to exit (for example, a diagnostic system requested the process controller to terminate).
RC_ABEND = 70	The process controller has had an unrecoverable problem and is forcibly terminating.
RC_QMUNAVAIL = 75	The process controller cannot continue because the queue manager for the process controller is unavailable.
RC_CONFIG = 78	The process controller cannot continue because there is a problem with the startup configuration data.

These exit codes are written to pceventX.log, where X can be any number, for example the log file name can be pcevent0.log.

Related reference

[Managed File Transfer diagnostic messages: BFGPC0001 - BFGPC9999](#)

Guidelines for transferring files

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

Read the relevant topics for further information.

Related reference

[“Transferring files and data sets between z/OS and distributed systems” on page 605](#)

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

[“Transferring between data sets on z/OS” on page 607](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“Transferring data sets to and from Connect:Direct nodes” on page 609](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[“Mappings between Connect:Direct process statement parameters and BPXWDYN keys” on page 611](#)

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

[“BPXWDYN properties you must not use with MFT” on page 616](#)

Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalOptions** property in the `agent.properties` file.

[“Transferring text files with MFT” on page 617](#)

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

[“Transferring text files between Connect:Direct and MFT” on page 619](#)

Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between a Managed File Transfer Agent and a Connect:Direct node.

[“Transferring files to or from protocol bridge agents” on page 620](#)

You can transfer files to and from an FTP or SFTP file server outside your Managed File Transfer network using a protocol bridge agent.

[“Transferring files to or from IBM i systems” on page 621](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

[“Transferring save files located in QSYS.LIB on IBM i” on page 625](#)

Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

[“Transferring generation data groups \(GDGs\)” on page 626](#)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

[“Using wildcard characters with MFT” on page 627](#)

You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

Transferring files and data sets between z/OS and distributed systems

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those attributes for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

Transferring from a file to a data set - binary transfers

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so that the data set is created properly. If you do not specify attributes, the system specifies the following default: a physical sequential data set with an undefined record format and the maximum block size (BLKSIZE) for the device (as returned by the DEVTYPE macro). For example, for DASD the size is 6144 and for tape the size is 32760. If you want to transfer a file on a distributed system to a z/OS data set in binary mode, note the following behavior:

Physical sequential (PS) destination data sets:

- The source file on the distributed system is read sequentially to fill each record or block.
- On variable format data sets, each record is filled to capacity.

Partitioned data set (PDS) destination data sets:

- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see [Object naming conventions](#). If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- The source file on the distributed system's is read sequentially to fill each record or block for the member.
- On variable format PDS members, each record is filled to capacity.

Transferring from a file to a data set - text transfers

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so the data set is created properly. If you want to transfer from a file on a distributed system to a z/OS data set as text, note the following behavior:

Physical sequential (PS) destination data sets:

- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in data set records (for non-ASA data sets only).
- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.
- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

PDS destination data sets:

- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see [Object naming conventions](#). If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in member records (for non-ASA data sets only).
- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.
- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

Transferring from a data set to a file - binary and text transfers

If you want to transfer from a data set to a file as binary or text, note the following behavior:

- The content of each record is transferred in binary form to a file; no record, block format information, or ASA format control characters are transferred.
- For text transfers only, each data set record becomes a line with text converted to the code page of the destination agent. That is, a carriage return-line feed (CRLF) is appended for a Windows destination system and carriage return (CR) is appended for a UNIX destination system.
- **Non-VSAM and PS source data sets.** The records for the source data set are transferred to the destination file and concatenated together. If the destination file exists, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer. If the destination is specified as a directory rather than a file, the destination filename will be the data set name excluding the high-level qualifier (HLQ).
- **PDS source data sets.** Each specified member, or all members if no member is specified, is extracted to the destination. If the destination specifies a directory, members are extracted to separate files. Otherwise each specified member is written to the destination file, resulting in effectively only one member being transferred. If the destination file exists for a member, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer.

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring between data sets on z/OS” on page 607](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Transferring between data sets on z/OS

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

Managed File Transfer does not support uncataloged data sets either on disk or tape. Existing data sets must be cataloged and new data sets will be cataloged.

Consider the following cases:

If you copy or move a data set between z/OS systems and the destination does not exist.

By default, the destination data set is created with the identical characteristics to the source. You can specify attributes for the destination data set to override the default characteristics. If you do this, a compatibility check is performed to ensure the transfer is possible.

If you copy or move a data set between z/OS systems and the destination already exists.

- If you specify attributes for the destination data set to override the default characteristics, a compatibility check is performed to ensure the destination data set can be accessed in the required way. However, you cannot override the following attributes:
 - Base data set organization and type
 - Logical record length (LRECL)
 - Block size (BLKSIZE)

If you are transferring a data set to tape.

When you transfer a data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

In addition, the user identifier that the destination agent is running as needs to have the correct authority to mount tapes. Refer to the documentation for the external security manager being used by your enterprise for information on how to do this.

If you are transferring from tape to a data set.

In order to access a dataset on tape, the user identifier that the source agent is running as needs to have the appropriate authority to mount tapes. Refer to the documentation for the external security manager being used by your enterprise for information on how to do this.

Data set compatibility

Review the following behavior and restrictions for data set compatibility:

Record format and length differences:

- Variable-format records use a 4 byte record length field in the record data. Therefore for a transfer from a fixed record to a variable record data set, the variable record length must be greater than or equal to the fixed record length plus 4. For a transfer from a variable format record data set to a fixed format record data set, the fixed format record data set record length must be greater than or equal to the variable record length minus 4.

Block size differences:

- For fixed- and variable-format record data, block size differences makes the source and destination data set layout different.
- For undefined format records, provided the destination block size is greater or equal to the source data set block size, you can transfer a data set.
- For undefined format data sets, you cannot transfer if the source block size is greater than the destination block size.

Partitioned data sets (PDS) and partitioned data set extended (PDSE) data sets

The following behavior and restrictions apply equally to PDS and PDSE:

- If you transfer a PDS or PDSE member to a destination PDS or PDSE, a member of the destination PDS or PDSE is created. If the destination PDS or PDSE member already exists, the member is overwritten. If you transfer a PDS or PDSE member to a non-PDS or non-PDSE destination data set, the destination data set is created to contain the member data. If the destination data set already exists, the data set is overwritten.
- If you attempt to transfer a PDS or PDSE to a non-PDS or non-PDSE destination, this results in all members of the PDS or PDSE being written to the non-PDSE destination. Each subsequent member transfer overwrites the previous contents of the non-PDSE destination or fails, depending on the transfer options.
- When you transfer a PDS or PDSE to a destination PDS or PDSE, a copy of the entire PDS or PDSE is created at the destination. If the destination PDS or PDSE already exists, members from the source are added. If a PDS or PDSE member already exists at the destination, the member is overwritten.
- The transfer of a non-PDS or non-PDSE to a destination PDS or PDSE, adds the contents of the non-PDS or non-PDSE as a new member of the PDS or PDSE. If the PDS member already exists, the member is overwritten. If you do not specify a name for a new member, a name is generated from the source data set or DD name.
- There is a known limitation with transfers to PDS and PDSE data sets on systems where disk space is limited. For more details, see the z/OS section in [Common MFT problems](#).

- **Note:** When you transfer a PDS or PDSE to a destination PDS or PDSE, the member information and statistics are not preserved. For example, if you transfer a load library that is stored as a PDS, the destination PDS is not usable as a load library.

Binary and text transfers

Binary transfer for data sets is defined as the record data in its binary form, as read from the data set using the default record format (type=record). Data is read and written on a record by record basis. The system service performs the necessary record and block conversion (where the data sets have different record and block settings) and the necessary ASA and machine control code conversion. If one data set is defined for ASA format control characters and the other is not appropriate, conversion to normal control codes is performed using the C/C++ system library function behavior.

Generation data groups (GDGs)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must already exist.

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring generation data groups \(GDGs\)” on page 626](#)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

[“Transferring data sets to and from Connect:Direct nodes” on page 609](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[“Transferring files and data sets between z/OS and distributed systems” on page 605](#)

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

Transferring data sets to and from Connect:Direct nodes

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

Specifying data set names

To specify a data set on a Connect:Direct node in a transfer request, use the syntax that is used for data set transfers between Managed File Transfer agents, but with two changes:

- You must prefix the data set name with the Connect:Direct node name and a colon (:). The syntax is as follows:

```
cdNode:data_set_name{;attrib1;...;attribN}
```

For example, to specify a partitioned data set called OBJECT.LIB on the system where the Connect:Direct node CD_NODE1 is located, use the following syntax:

```
CD_NODE1: //'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)
```

In this example, three optional attributes are specified by the text
RECFM(F,B);BLKSIZE(800);LRECL(80).

- The specified data set name is interpreted as a fully qualified data set name, regardless of whether it is enclosed by single quotation mark characters. The system never adds any prefix. If you want to specify a prefix, such as the user ID that the agent runs under, you must specify it as part of the data set name. This differs from the behavior for data set transfers that involve only Managed File Transfer agents, where if the specified data set name is not enclosed by single quotation mark characters, the system adds a prefix of the default high-level qualifier for the destination agent.

Except for these two changes, specify the data set name and any optional attributes using the same syntax that is used for data set transfers between Managed File Transfer agents, which has the following rules:

- You must prefix the data set name with two forward slash characters (//).
- If you want to specify data set attributes, provide these after the data set name, separated by semicolons. Attributes must be provided in the format *key(value)*, which is suitable for BPXWDYN.

For more information about specifying data sets in a transfer request, see [“fteCreateTransfer: start a new file transfer”](#) on page 417 and [“fteCreateTemplate: create new file transfer template”](#) on page 401.

Parameters to use in your transfer request

For most transfer requests that involve data sets on Connect:Direct nodes, you can specify the source and destination data sets in the same way as you would for a data set transfer that involves only Managed File Transfer agents. Use the **source_specification**, **-ds**, and **-dp** parameters with the **fteCreateTransfer** or **fteCreateTemplate** commands. This syntax is supported for the following scenarios:

- All the agents involved in the transfer are IBM WebSphere MQ File Transfer Edition 7.0.4 or later
- The source agent is the Connect:Direct bridge agent, and is therefore IBM WebSphere MQ File Transfer Edition 7.0.4 or later, and the destination agent is IBM WebSphere MQ File Transfer Edition 7.0.3 or earlier

If the destination agent is the Connect:Direct bridge agent, and the source agent is IBM WebSphere MQ File Transfer Edition 7.0.3 or earlier, you must make the following changes to your transfer request:

- To specify a sequential data set or partitioned data set (PDS) member as the destination of a transfer, use the **-df** parameter.
- To specify a PDS as the destination of a transfer, use the **-dd** parameter.

You can also use this syntax as an alternative to the usual **-ds** and **-dp** parameters for transfers where the source agent is IBM WebSphere MQ File Transfer Edition 7.0.4 or later. For example, if you want to use a consistent syntax across all your scenarios and some scenarios involve a source agent that is IBM WebSphere MQ File Transfer Edition 7.0.3 or earlier, use the **-df** and **-dd** parameters.

Note: If the destination of the transfer is a PDS and the destination agent is the Connect:Direct bridge agent, you must specify the **-de** parameter with the value of `overwrite`.

Specifying data set attributes

Certain data set attributes are set by Managed File Transfer and passed through as parameters to the Connect:Direct **COPY** process. You can also supply certain attributes in the transfer request, by specifying the appropriate BPXWDYN key. The Connect:Direct bridge converts keys that have equivalent Connect:Direct properties to the format that is required by Connect:Direct. For example, in the data set specification `CD_NODE1: // 'OBJECT.LIB' ;RECFM(F,B);BLKSIZE(800);LRECL(80)`, the attributes `RECFM(F,B);BLKSIZE(800);LRECL(80)` are converted to `DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)`.

For details of the mappings between these two types of parameter, including details of the BPXWDYN keys that are supported for use with a Connect:Direct transfer, see [“Mappings between Connect:Direct process statement parameters and BPXWDYN keys”](#) on page 611. Not all BPXWDYN keys have an

equivalent Connect:Direct process parameter, and not all Connect:Direct process parameters have an equivalent BPXWDYN key.

Additional considerations

- If your transfer destination is a partitioned data set at a Connect:Direct node, you must create the partitioned data set before the transfer, because the Connect:Direct node does not create it for you.

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related tasks

 [Transferring a data set to a Connect:Direct node on z/OS](#)

You can transfer a data set from a Managed File Transfer agent on z/OS to a Connect:Direct node on z/OS by using a Connect:Direct bridge that is located on a Windows or Linux system.

Related reference

[“Transferring between data sets on z/OS” on page 607](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“fteCreateTemplate: create new file transfer template” on page 401](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[“Connect:Direct file paths specified with a double forward slash” on page 331](#)

If, as part of a file transfer, you specify a file located on a Connect:Direct node by using a file path that starts with a double forward slash (//), the file is treated as a data set.

Mappings between Connect:Direct process statement parameters and BPXWDYN keys

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

For more information about IBM Sterling Connect:Direct process statements, see the [Connect:Direct Process Language Reference Guide](#).

<i>Table 58. Parameters to the Connect:Direct COPY statement, and the equivalent BPXWDYN keys used by Managed File Transfer</i>	
Parameter to Connect:Direct COPY statement	BPXWDYN key
DSN	DSN (valid for transfers to and from data sets). Specifying this key overrides the parameter value that is assigned by Managed File Transfer, which is based on the source or destination file specifications that are provided in the transfer request.
FILE	No mapping for data sets.

Table 58. Parameters to the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Parameter to Connect:Direct COPY statement	BPXWDYN key
PNODE	No mapping. The primary node for the transfer is identified by Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
SNODE	No mapping. The secondary node for the transfer is identified by Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
DCB	See Mappings for subparameters of DCB
DISP	See Mappings for subparameters of DISP for a COPY From statement and Mappings for subparameters of DISP for a COPY To statement
RESGDG	No mapping
LABEL	See Mappings for subparameters of LABEL
MSVGP	No mapping
UNIT	UNIT
VOL	See Mappings for subparameters of VOL
ALIAS	No mapping
EXCLUDE	No mapping
PDS.DIR	No mapping. Managed File Transfer sets the value of this process parameter to N, so no user-related information that is in the directory is sent.
REPLACE NOREPLACE	No BPXWDYN equivalent. The behavior when a destination data set already exists on the destination system is defined by the value of the -de (destination_file_behavior) parameter in the transfer request. For more information about the default behavior of Managed File Transfer when a destination data set already exists, see “ Transferring between data sets on z/OS ” on page 607.
SELECT	No BPXWDYN equivalent. The data set members that are selected for copying are defined by the source file specification in the transfer request.
BUFND	No mapping
IOEXIT	No mapping
DATAEXIT	No mapping
SYSOPTS	See Mappings for subparameters of SYSOPTS
TYPE	No mapping
AVGREC	No mapping
DATACLAS	DATACLAS

Table 58. Parameters to the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Parameter to Connect:Direct COPY statement	BPXWDYN key
DSNTYPE	DSNTYPE. Specifying a value of PDS for this key overrides the parameter value that is assigned by Managed File Transfer, which is LIBRARY. There are no mappings for any other value - EXTPREF, EXTREQ, BASIC, or LARGE. Specifying any of these unsupported values produces an error. Specifying PDS or LIBRARY for a sequential data set produces an error.
KEYLEN	No mapping
KEYOFF	No mapping
LIKE	LIKE
LRECL	No mapping
MGMTCLAS	MGMTCLAS
RECORG	No mapping
SECMODEL	No mapping
STORCLAS	STORCLAS
SPACE	See Mappings for subparameters of SPACE
SYSOUT	No mapping
CKPT	No mapping
COMPRESS	No mapping
SECURE	No mapping

Table 59. Subparameters of the **DCB** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DCB parameter	BPXWDYN key
model-file-name	No mapping
BLKSIZE	BLKSIZE
NCP	BUFNO
DEN	No mapping
DSORG	DSORG
KEYLEN	No mapping
LIMCT	No mapping
LRECL	LRECL
OPTCD	No mapping
RECFM	RECFM
RKP	No mapping
TRTCH	TRTCH

Table 60. Subparameters of the **DISP** parameter for the Connect:Direct **COPY From** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DISP parameter for a COPY From statement	BPXWDYN key	Details
[OLD SHR]	[OLD SHR]	Specifies the status of the data set before the transfer. Managed File Transfer sets this subparameter to SHR .
[KEEP DELETE]	[KEEP DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. The value set by Managed File Transfer depends on the source file disposition, defined by the -sd parameter.
[KEEP DELETE]	[KEEP DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. Managed File Transfer sets this subparameter to KEEP .

Table 61. Subparameters of the **DISP** parameter for the Connect:Direct **COPY To** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DISP parameter for a COPY To statement	BPXWDYN key	Details
[NEW OLD MOD RPL SHR]	[NEW OLD MOD SHR]	Specifies the status of the data set before the transfer. The value set by Managed File Transfer depends on the value of the -de (destination_file_behavior) parameter in the transfer request. If the destination data set does not already exist, the subparameter value is NEW . If the data set already exists, the subparameter value is RPL . Managed File Transfer does not support the key RPL being provided in a transfer request.
[KEEP CATLG]	[KEEP CATLOG] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. Managed File Transfer sets this subparameter to CATALOG .
[KEEP CATLG DELETE]	[KEEP DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. Managed File Transfer sets this subparameter to DELETE .

Table 62. Subparameters of the **LABEL** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the LABEL parameter for a COPY statement	BPXWDYN key	Details
file-sequence-number	SEQUENCE	
[SL AL BLP LTM NL]	LABEL(<i>type</i>)	The possible values of <i>type</i> are NL, SL, NSL, SUL, BLP, LTM, AL, and AUL. Connect:Direct accepts a subset of these values. If you specify a value that is not supported by Connect:Direct, Connect:Direct produces an error message.
[PASSWORD NOPWREAD]	No mapping	
[IN OUT]	No mapping	
[RETPD EXPDT]	RETPD	EXPDT not supported

Table 63. Subparameters of the **VOL** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the VOL parameter for a COPY statement	BPXWDYN key
PRIVATE	No mapping
RETAIN	No mapping
volume-sequence-no	No mapping
volume-count	MAXVOL
SER	VOL
REF	No mapping

Table 64. Subparameters of the **SYSOPTS** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the SYSOPTS parameter for a COPY statement	BPXWDYN key
DBCS	No mapping
CODEPAGE	Value is dependent on Managed File Transfer transfer options. For more information, see “Transferring text files with MFT” on page 617.
DATATYPE	No mapping. Managed File Transfer sets this value to TEXT for text transfers to or from a data set, and otherwise to BINARY.
XLATE	No mapping. Managed File Transfer sets this value to NO when the value of DATATYPE is TEXT.
STRIP.BLANKS	No mapping. Managed File Transfer sets this value to YES when the value of DATATYPE is TEXT.
PERMISS	No mapping

Table 64. Subparameters of the **SYSOPTS** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Subparameters of the SYSOPTS parameter for a COPY statement	BPXWDYN key
PRECOMP	No mapping
UNIQUE	No mapping
SYSOUT	No mapping

Table 65. Subparameters of the **SPACE** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the SPACE parameter for a COPY statement	BPXWDYN key
CYL	CYL
TRK	TRACKS
blk	BLOCKS
av-rec-len	No mapping
prim, [sec], [dir]	SPACE(prim[,sec]), DIR
RLSE	RELEASE
CONTIG	No mapping
ROUND	No mapping

Related concepts

“The Connect:Direct bridge” on page 220

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

Related tasks

 Transferring a data set to a Connect:Direct node on z/OS

You can transfer a data set from a Managed File Transfer agent on z/OS to a Connect:Direct node on z/OS by using a Connect:Direct bridge that is located on a Windows or Linux system.

Related reference

Transferring data sets to and from Connect:Direct nodes

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

BPXWDYN properties you must not use with MFT

Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalOptions** property in the agent.properties file.

There are a number of BPXWDYN options that must not be specified with Managed File Transfer because they are used by the agent or they are not supported. If you use these options they can cause unpredictable behavior; the options are listed in the following table.

BPXWDYN options	Description
DA DSN	Specifies the data set name to allocate.

BPXWDYN options	Description
FI DD	Specifies the ddname to allocate.
FILEDATA	Specifies, to the sequential access method services, whether the data is treated as text or binary.
OLD SHR MOD NEW SYSOUT	Specifies the data set status.
REUSE	Specifies that the named data set is freed before the function is performed.
HOLD	Specifies that the output data set is to be held until released by the user or operator.
KEEP DELETE CATALOG UNCATALOG	Specifies the data set disposition after it is freed.
RECORGL(S)	Creates a VSAM linear data set.
MSG	Directs allocation messages. Note: This option can be used, but because Managed File Transfer uses this option to direct error information to the transfer log, using it can cause unpredictable behavior.

Transferring text files with MFT

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

Unless you specify otherwise, conversion is from the default code page of the file's source system to the default code page of its destination system. Additionally, text file transfer performs new line conversion, which means that new line characters for the destination file are those native to its destination platform. You can override the use of the default code pages on a system by specifying the code page to use for reading the source file and writing the destination file. You can also specify the end-of-line character sequence to use for the destination file. For more information, see the topics [“fteCreateTransfer: start a new file transfer”](#) on page 417 and [“Using transfer definition files”](#) on page 152.

Text file transfers perform simple code point substitutions between code pages. Text file transfers do not perform complex transfers or translations of data, for example, conversions between visual and logical forms of bidi data or text shaping.

Table 66. Text file transfer behavior for all platforms

Area	Default behavior	Can you change this behavior?
Source file encoding	Source platform encoding	Yes When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.
Source file end of line character sequence	Convert a single (LF) or (CRLF) sequence to the destination end of line character sequence	No

Table 66. Text file transfer behavior for all platforms (continued)

Area	Default behavior	Can you change this behavior?
Destination file encoding	Destination platform encoding	Yes When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.
Destination file end of line character sequence	Destination platform EOL	Yes
Text replacement character sequence for unmappable or malformed characters in the source or destination	Blank, meaning the transfer fails if unmappable characters or malformed characters are present. You can use the <code>textReplacementCharacterSequence</code> property to specify the replacement text, which is described in “The MFT agent.properties file” on page 526.	Yes

z/OS data sets



When data set records are accessed in text mode, each record represents a single line. New line characters do not exist in the record but for ASA format data sets an ASA format control code character is set that represents a new line (or other control character). When a line of text with a terminating new line character is written to a record, the new line character is either automatically removed or an appropriate ASA control code is set, as appropriate. When a record is read a new line character is automatically appended to the return data. For ASA format data sets this character can be multiple new lines or a form feed, as appropriate for the ASA control code of the record.

Additionally, for fixed-format data sets when a record is read the new line is appended after the last character in the record that is not a space character, thus making fixed-format data sets suitable for storing text.

Table 67. Additional text file transfer behavior specific to z/OS

Area	Default behavior	Can you change this behavior?
Maximum line length	Destination data set LRECL or BLKSIZE setting, as appropriate	No
Wrap over length lines	Wrap. The line is split over multiple records and blocks as required.	No

When the Managed File Transfer agent is run, the environment variable `_EDC_ZERO_RECLLEN` is always set to "Y". This setting makes Managed File Transfer text transfer behavior the same as FTP for variable and fixed block data sets. However, for undefined format data sets, Managed File Transfer converts single space lines to an empty line and preserves empty lines. FTP converts empty lines to single space lines

and preserves single space lines. Table 3 describes the Managed File Transfer behavior and how FTP behavior differs.

The format of the data set also determines how each line of text is written to a record. For non-ASA format data sets newline and carriage-return characters are not written to the record. For ASA format data sets, the first byte of each record is an ASA control code representing end of lines, a form feed, and other codes, as appropriate. Because ASA control codes are at the start of each record, if the source text file does not start with a new line character sequence, a blank (' ') ASA control character sequence (which equates to a newline) is inserted. This means that if the ASA data set is transferred to a file, a blank line is present at the start of the file.

Data set format	Original text line in file	Data set record	Read of data set record	FTP Read behavior
Fixed block	Empty line	Space filled record	Empty line	Same as MFT
Fixed block	Single space	Space filled record	Empty line	Same as MFT
Variable block	Empty line	Empty record	Empty line	Same as MFT
Variable block	Single space	Single space record	Single space	Same as MFT
Undefined	Empty line	Single space record	Empty line	Single space
Undefined	Single space	Single space record	Empty line	Single space

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring text files between Connect:Direct and MFT” on page 619](#)

Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between a Managed File Transfer Agent and a Connect:Direct node.

[“Available code pages for MFT” on page 657](#)

This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by Managed File Transfer.

Transferring text files between Connect:Direct and MFT

Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between a Managed File Transfer Agent and a Connect:Direct node.

For information about the behavior of text transfers in Managed File Transfer, see [“Transferring text files with MFT” on page 617](#).

- Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.
 - If your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list.
 - If the remote node is on a Windows system, select Windows.
 - If the remote node is on a UNIX or Linux system, select UNIX.

-  If the remote node is on a z/OS system, select OS/390.

Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

- Ensure that for each remote node you transfer a file to or from, you specify the operating system type of the remote Connect:Direct node in the `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory. For more information, see [“Configure the ConnectDirectNodeProperties.xml file to include information about the remote Connect:Direct nodes”](#) on page 137 and [“Connect:Direct node properties file format”](#) on page 563.

Connect:Direct uses the network map information to determine which line ending to use.

- If the destination of a transfer is a Managed File Transfer Agent, this agent performs the line ending conversion.
- If the destination of a transfer is a Connect:Direct node, the Connect:Direct bridge agent performs the line ending conversion.

Related reference

[“Transferring text files with MFT”](#) on page 617

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

[“What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly”](#) on page 330

When you transfer files in text mode between an MFT agent and a Connect:Direct node, code page and end-of-line character conversion is performed. The transfer uses the operating system information in the network map of the Connect:Direct bridge node to determine the end-of-line characters of a remote node. If the information in the network map is incorrect, the end-of-line character conversion might be performed incorrectly.

Transferring files to or from protocol bridge agents

You can transfer files to and from an FTP or SFTP file server outside your Managed File Transfer network using a protocol bridge agent.

When you transfer files using the protocol bridge, the bridge must have permission to read the source or destination directory containing the files you want to transfer. For example, if you want to transfer files from the directory `/home/fte/bridge` that has only execute permissions (`d--x--x--x`), any transfers you attempt from this directory fail with the following error message:

```
BFGBR0032E: Attempt to read filename from the protocol file server has failed with server error 550
Failed to open file.
```

During file transfer, files are typically written as temporary files at the destination and are then renamed when the transfer is complete. However, if the transfer destination is a protocol file server that is configured as limited write (users can upload files to the protocol file server but cannot change those uploaded files in any way; effectively users can write once only), transferred files are written to the destination directly. This means that if a problem occurs during the transfer, the partially-written files remain on the destination protocol file server and Managed File Transfer cannot delete or edit these files. In this situation the transfer fails.

Ensure that you have another agent in your Managed File Transfer network in addition to the protocol bridge agent. The protocol bridge agent is a bridge to the FTP or SFTP server only and does not write transferred files to the local disk. If you want to transfer files to or from the FTP or SFTP server you must use the protocol bridge agent as the destination or source for the file transfer (representing the FTP or SFTP server) and another standard agent as the corresponding source or destination.

Managed transfer requests that require a new directory to be created on an SFTP file server

Managed File Transfer protocol bridge agents use the third-party JSch library to communicate with file servers using the SFTP protocol. If the protocol bridge agent attempts to transfer a file into a directory that does not exist on a file server and JSch is unable to perform the requested SFTP operation to create that directory, because the user that the protocol bridge agent logs into the file server with does not have permission to do so, JSch throws an exception back to the protocol bridge agent. The protocol bridge agent then marks the managed transfer as "Failed" and generates a supplementary message. If JSch has provided more information about the failure, the protocol bridge agent includes this information in the supplementary message:

```
BFGTR0072E: The transfer failed to complete due to the exception:  
BFGBR0119E: Bridge agent was unable to create directory directory name because message from JSch  
exception
```

V 9.0.0.9 From IBM MQ 9.0.0 Fix Pack 9, if the JSch exception does not contain any more information about the failure, the protocol bridge agent generates the following supplementary message:

```
BFGTR0072E: The transfer failed to complete due to the exception:  
BFGBR0209E: Bridge agent was unable to create directory directory name
```

Related concepts

[“The protocol bridge” on page 204](#)

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

IBM i *Transferring files to or from IBM i systems*

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

Each file on an IBM i system is tagged with a coded character set ID (CCSID) value that identifies the data encoding of the file. For example, a file containing EBCDIC data might have a CCSID value of 037 and a file containing ASCII data might have a CCSID value of 819.

For text mode transfers, Managed File Transfer converts data when there are file encoding differences between source and destination files. However, Managed File Transfer currently ignores CCSID tags associated with files on IBM i systems. Instead, it uses the JVM file encoding property of the JVMs running the source agent and destination agent. The default value of this property is based on locale (but you can override this default on your IBM i system using the `SystemDefault.properties` file described in the following section: [“Changing the file.encoding record in the SystemDefault.properties file” on page 622](#)). With this default implementation, an agent that transfers files in text mode is limited in its ability to handle text files with different file encodings. For example, you cannot use the same agent to transfer files containing EBCDIC text and also files containing ASCII text without stopping and restarting the agent with the appropriate (that is, EBCDIC or ASCII) file encoding override in place. On IBM i V6R1 systems, you can check the file encoding value of the JVM that is running the agent job by using WRKJVMJOB, option 7 to Display Current® Java System Properties. (The WRKJVMJOB command does not exist on IBM i V5R4 systems.)

If you plan to use Managed File Transfer to transfer text files with different file encodings, consider creating multiple agents and multiple users who start those agents, so that each unique encoding has an agent that is ready and enabled to transfer that type of data.

For example, if you want to transfer a file containing EBCDIC text with CCSID value of 037 from an IBM i system (source) to another IBM i V6R1 system (destination) where you want the file content at the destination to be converted to ASCII text with CCSID value of 819, complete the following steps:

1. Select a source agent with a JVM file encoding of Cp037.
2. Select a destination agent with a JVM file encoding of ISO8859_1.

3. Select text mode transfer, and other specifications as needed.

Changing the file.encoding record in the SystemDefault.properties file

To enable a JVM running an agent for a particular encoding, complete the following steps:

1. Determine which user starts the agent that runs on the IBM i system. This is the agent that services the Managed File Transfer file transfer request.

Create a `SystemDefault.properties` file in the home directory of that user as needed. For example, if you start the agent, use Qshell to run the following command:

```
touch -C 819 /home/your_userID/SystemDefault.properties
```

2. Using Qshell, run the `/qibm/proddata/mqm/bin/fteStopAgent` command to stop the agent as needed.
3. Update the `SystemDefault.properties` file that is described in step 1 to ensure that the file contains a record like the following:

```
file.encoding=java_encoding
```

where *java encoding* corresponds to the type of data that is contained in the file, and matches a `file.encoding` value from the following table: [File.encoding values and System i5® CCSID](#).

4. The user identified in step 1 must complete the following steps:
 - a. On IBM i V5R4 only: add the `QIBM_PASE_DESCRIPTOR_STDIO` environment variable (*JOB scope) to 'B' if using EBCDIC file encoding, or 'T' if using ASCII encoding. For example:

```
ADDENVVAR ENVVAR('QIBM_PASE_DESCRIPTOR_STDIO') VALUE('B') REPLACE(*YES)
```

- b. If Qshell is active, press **F3=Exit** to end Qshell.
- c. Start Qshell and run the `/qibm/proddata/mqm/bin/fteStartAgent` command as appropriate to restart the agent.

When the file encoding of the JVM running the agent has been changed, the agent log is written with that encoding. If you want to read the contents of the agent log, you must use a viewer that is enabled for that encoding.

Using a transfer definition for data conversion

An alternative way to convert data when files are being transferred is to create a transfer definition that specifies file encoding, or to use the `-sce` and `-dce` parameters of the `fteCreateTransfer` command. If you use these parameters when the destination is an IBM i system, this can result in files that have incorrect CCSID tags. For this reason, the recommended approach for controlling data conversion with files that are located on IBM i systems is to use `SystemDefault.properties` as described in the preceding section.

Protocol bridge limitation

On IBM i, you cannot transfer EBCDIC files to or from an SFTP server using a protocol bridge agent.

Related tasks

[Installing IBM MQ server on IBM i](#)

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring save files located in QSYS.LIB on IBM i” on page 625](#)

Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

IBM i *Transferring physical files located in QSYS.LIB on IBM i*

Managed File Transfer supports the transfer of physical file members in the QSYS.LIB file system between two IBM i systems. Consider the following information when you request file transfers of physical file members.

A physical file member on IBM i is located in a physical file, which in turn is located in a library on IBM i. A library can be one of the standard libraries that ship with the operating system (for example, QSYS or QGPL) or it can be a library that you have created.

Physical files in the QSYS.LIB file system are identified in two different ways on IBM i. When you run CL commands on an IBM i command line, use the following naming syntax:

```
FILE(library name/file name) MBR(member name)
```

For example, a physical file member that is called MYMBR is in a file that called MYFILE in a library that is called SOMELIB is identified as FILE(SOMELIB/MYFILE) MBR(MYMBR). You can also identify the same physical file member by specifying a UNIX-like path name that follows the Integrated File System (IFS) naming convention. Using the IFS naming convention, MYMBR in MYFILE in SOMELIB has the following path name:

```
/QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR.MBR
```

For more information, see [Path names in the QSYS.LIB file system](#).

Managed File Transfer on IBM i recognizes the IFS naming convention but does not support the syntax used by CL commands. The following examples illustrate valid and invalid path names for MFT. The following example is a valid path name for a physical file member:

```
/QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR.MBR
```

This example assumes MYFILE is a physical file in the library SOMELIB and contains a member that is called MYMBR.

The following examples are invalid path names for physical file member transfers:

- /QSYS.LIB/SOMELIB.LIB/MYFILE.FILE (.FILE assumes a SAVF, not a physical file. If MYFILE is a physical file, the transfer fails with an invalid file type error)
- /QSYS.LIB/MYLIB.LIB/ (physical file and member names are required)
- /QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR (the member name must contain an extension of .MBR)
- /QSYS.LIB/SOMELIB.LIB/MYFILE/MYMBR.MBR (the physical file name extension must be .FILE)

Transferring multiple physical file members from a physical file in a single transfer request

Managed File Transfer on IBM i supports the transfer of multiple physical file members from a single physical file as a single transfer request. You can specify an appropriate path name that includes wildcard characters as shown in the following examples:

- ABCLIB contains a physical file MYFILE with multiple members. To transfer all these members in a single request, specify the following path name: /QSYS.LIB/ABCLIB.LIB/MYFILE.FILE/* .MBR
- XYZLIB contains a physical file MYFILE whose member names differ by a single character, that is: TEST1.MBR, TEST2.MBR, TEST3.MBR, and so on. To transfer all these members in a single request, specify the following path name: /QSYS.LIB/XYZLIB.LIB/MYFILE.FILE/TEST? .MBR.

The following types of transfer requests are not supported for transferring multiple physical file members and result in an error:

- /QSYS.LIB/MYLIB.LIB/*.*
- /QSYS.LIB/MYLIB.LIB/*
- /QSYS.LIB/MYLIB.LIB/*.FILE/MYMBR.MBR
- /QSYS.LIB/MYLIB.LIB/MYFILE*.FILE/*.MBR (there is no support for wildcarding on file names, only on member names)
- /QSYS.LIB/MYLIB.LIB/*.FILE/*.MBR
- /QSYS.LIB/MYLIB.LIB/MYFILE.FILE (.FILE assumes a SAVF not a physical file, so if MYFILE is a physical file, the transfer fails with invalid file type error)

Transferring physical file members to and from non-IBM i systems

MFT supports the transfer of physical file members to and from non-IBM i systems, such as UNIX, Linux, and Windows. All transfers must be done in text mode. The following examples illustrate some of the supported **fteCreateTransfer** requests when working with non-IBM i systems:

- This command transfers physical file member FILE(FROMIBMI/FILE1) MBR(FILE1) on IBM i to text file /home/qfte/fromibmi/linux.mbr.txt on Linux:

```
fteCreateTransfer -da linux -dm QM1 -sa ibmi -sm QM1 -t text -df /home/qfte/fromibmi/
linux.mbr.txt /qsys.lib/fromibmi.lib/file1.file/file1.mbr
```

- This command transfers physical file member FILE(FROMIBMI/FILE1) MBR(FILE1) on IBM i to text file C:\FTE\fromibmi\windows.mbr.txt on Windows:

```
fteCreateTransfer -da windows -dm QM1 -sa ibmi -sm QM1 -t text -df
C:\FTE\fromibmi\windows.mbr.txt /qsys.lib/fromibmi.lib/file1.file/file1.mbr
```

- This command transfers text file C:\FTE\toibmi\file.txt on Windows to physical file member FILE(TOIBMI/EXISTS) MBR(WINDOWS) on IBM i:

```
fteCreateTransfer -da ibmi -dm QM1 -sa windows -sm QM1 -t text -df /qsys.lib/toibmi.lib/
exists.file/windows.mbr C:\FTE\toibmi\file.txt
```

The following commands are examples of invalid physical file member transfers with non-IBM i systems:

- This command fails because the source file on Windows has a .txt file extension but a destination directory of .file has been specified. When transferring using the destination directory parameter to specify a destination physical file, the source file extension must be .mbr file, for example, C:\FTE\toibmi\file.mbr

```
fteCreateTransfer -da ibmi -dm QM1 -sa windows -sm QM1 -t text -dd /qsys.lib/toibmi.lib/
windows.file C:\FTE\toibmi\file.txt
```

- The default transfer mode is binary and text mode must be specified when transferring physical file members.

```
fteCreateTransfer -da windows -dm QM1 -sa ibmi -sm QM1 -df C:\FTE\fromibmi\file.bin /qsys.lib/
fromibmi.lib/file1.file/file1.mbr
```

MFT supports the transfer of physical file members that are in the QSYS.LIB file system but does not support the transfer of source physical file members that are in the QSYS.LIB file system. File transfers in the QDLS file system are supported using the provided sample user exits. You can use the user exit samples provided in MFT for the following tasks:

- Transfer files in the QDLS file system.
- Automatically transfer physical file members from an IBM i library in the same way as an MFT file monitor.
- Delete an empty file object when the source file member is deleted as part of the transfer.

For more information, see [“Sample MFT on IBM i user exits” on page 250](#).

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring files to or from IBM i systems” on page 621](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

IBM i **Transferring save files located in QSYS.LIB on IBM i**

Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

A save file on IBM i is located in a library on IBM i. A library can be one of the standard libraries that ship with the operating system for example QSYS or QGPL or it can be a library that is created by the user. Save files in the QSYS.LIB file system are identified in two different ways on IBM i. When working with CL commands on an IBM i command line, the naming syntax used is as follows:

```
FILE(library name/file name)
```

For example, a save file called MYSAVF is located in a library called SOMELIB is identified as FILE(SOMELIB/MYSAVF).

You can also identify the same save file by specifying a UNIX-like path name that follows the Integrated File System (IFS) naming convention. See [Path names in the QSYS.LIB file system](#) for more information. Using the IFS naming convention, MYSAVF in SOMELIB has the following path name:

```
/QSYS.LIB/SOMELIB.LIB/MYSAVF.FILE
```

Managed File Transfer on IBM i recognizes the IFS naming convention but does not support the syntax used by CL commands. The following examples illustrate valid and invalid path names for Managed File Transfer.

Some examples of valid path names for save file transfers are as follows:

- /QSYS.LIB/SOMELIB.LIB/MYSAVF.FILE (assuming MYSAVF save file is located in the library SOMELIB)
- /QSYS.LIB/MYSAVF.FILE (assuming MYSAVF is located in the library QSYS)

Some examples of invalid path names for save file transfers are as follows:

- SOMELIB.LIB/MYSAVF.FILE (Path name must start with /QSYS.LIB)
- /QSYS.LIB/MYLIB.LIB (Path must end in a save file name, not a library name)
- /QSYS.LIB/MYLIB.LIB/ (Save file name is required)
- /QSYS.LIB/SOMELIB.LIB/MYSAVF (Save file name must have a .FILE extension in name)
- /QSYS.LIB/SOMELIB.LIB/MYSAVF.SAVF (Save file name extension must be .FILE)

Transferring multiple save files from a library in a single transfer request

Managed File Transfer on IBM i supports the transfer of multiple save files from a library as a single transfer request. You can specify an appropriate path name that includes wildcard characters as shown in the following examples:

- ABCLIB contains many save files. To transfer all these files in a single request, specify the following path name:

```
/QSYS.LIB/ABCLIB.LIB/*.FILE
```

- XYZLIB contains several save files whose names differ by a single character, that is: TEST1.FILE, TEST2.FILE, TEST3.FILE, and so on. To transfer all of these files in a single request, specify the following path name:

```
/QSYS.LIB/XYZLIB.LIB/TEST?.FILE
```

The following types of transfer requests are not supported for transferring multiple save files and result in an error:

- ```
/QSYS.LIB/MYLIB.LIB/*.*
```

- ```
/QSYS.LIB/MYLIB.LIB/*
```

Managed File Transfer supports the transfer of save files that are located in the QSYS.LIB file system but the transfer of other types of files that are located in the QSYS.LIB file system is not supported. However, Managed File Transfer provides samples that use the save file support and use predefined `fteAnt` tasks to demonstrate how a complete library, a source physical file, or database file can be transferred between two IBM i systems. See [“Getting started using Ant scripts with MFT” on page 242](#) for details on how to customize and use these samples.

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring files to or from IBM i systems” on page 621](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

Transferring generation data groups (GDGs)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

Note: When creating a GDG entry in a batch environment using `BASEGDG(+n)`, it cannot be referred to later in the same job by using the same positive generation number. Maintaining the same GDG entry numbers between steps of a job is a function of JCL and is not available to utility functions that update the GDG by using dynamic allocation. Therefore, a job that creates a new generation using `BASEGDG(+1)` would find the GDG updated as soon as the transfer successfully completes and would then need to refer to the same data set as `BASEGDG(0)`.

GDG examples

The following examples show the `fteCreateTransfer` command using GDGs. In the examples, the name `BASEGDG` refers to an existing base GDG name. The name `DSET` refers to a sequential data set that is to be created. The name `/u/user/file.dat` refers to the name of a source data file.

This command copies `file.dat` into a new generation in `BASEGDG`. The absolute name of the new generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG(+1)" /u/user/file.dat
```

This command copies file .dat into the generation with the absolute name specified in BASEGDG:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG.G0009V00" /u/user/file.dat
```

This command copies the most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(0)"
```

This command copies the next most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(-1)"
```

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“Transferring between data sets on z/OS” on page 607](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

Using wildcard characters with MFT

You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

Multiplatforms

You can use the following wildcard characters on [Multiplatforms](#):

?

Use the question mark (?) to represent exactly one character. All of the other characters specified are required in matching file names.

For example, ab?.jpg matches the files abcd.jpg, abed.jpg, and abfd.jpg.

*

Use the asterisk character (*) to represent zero or more characters.

For example *.txt matches the files abc.txt and x.txt, but not newtxt because the period (.) in the file names is a required character.

The pattern *txt matches the files abc.txt, x.txt, and newtxt.

You must enclose the asterisk character (*) in double quotation marks. If you do not, the character will be interpreted by the command shell and might cause the command to fail.

  On UNIX and Linux, using the asterisk character (*) will not include the pseudo hidden files, for example .bashrc.

If the operating system is case-insensitive for file and path names, for example Windows, the pattern match is case-insensitive. You can use wildcard characters to specify file names only: you cannot use wildcards in directory names.

Protocol bridge agent

If you are using a protocol bridge agent to transfer files from an FTP, FTPS, or SFTP file server, wildcard matching is case sensitive, regardless of the platform that the file server is actually running on.

Connect:Direct bridge

When the source of a transfer is a Connect:Direct bridge agent that is requesting files from a Connect:Direct node, wildcards are not supported.

IBM i



You can use the following wildcard characters on IBM i platforms:

?

Use the question mark (?) to represent exactly one character. All of the other characters specified are required in matching file names.

For example, ab?d . jpg matches the files abcd . jpg, abed . jpg, and abfd . jpg.

Use the asterisk character (*) to represent zero or more characters.

For example * . txt matches the files abc . txt and x . txt.

The pattern *txt matches the files abc . txt, x . txt, and newtxt because the period (.) in the pattern is a required character.

For additional considerations regarding the use of wildcard characters with save file transfers, see [Transferring save files that reside in QSYS.LIB file system on IBM i systems](#) .

z/OS



For z/OS systems the wildcard character rules for Managed File Transfer follow the standard ISPF wildcard conventions in general. There are specific rules for both sequential and partitioned data sets as follows:

Sequential data sets



When you reference sequential data sets, you can use data set name qualifiers containing asterisks (*) and percent signs (%) as follows:

Use a single asterisk (*) to represent at least one qualifier. A single asterisk within a qualifier represents zero or more characters.

Use double asterisks (**) to represent zero or more qualifiers. You cannot use a double asterisk within a qualifier.

%

Use a single percent sign (%) to represent one single alphanumeric or national language character.

%%

Use between one and eight percent signs to represent zero or more characters.

Partitioned data sets



When you reference partitioned data sets, you can specify wildcard characters for the member names only. You can use data set name qualifiers containing asterisks (*), underscores (_), and question marks (?) as follows:

*

Use the asterisk (*) character to represent zero or more characters.

_

Use the underscore (_) character to represent exactly one character.

?

Use the question mark (?) character to represent exactly one character. The question mark is an alternative to the underscore character and is provided as an addition to ISPF conventions.

Directories

By default if you create a file transfer with a wildcard pattern that matches subdirectories, the subdirectories are not transferred. You can specify the **-r** parameter on the `fteCreateTransfer` command to include subdirectories that match the wildcard pattern. When you transfer a subdirectory, the entire contents and structure of the subdirectory are transferred: including all of its files, subdirectories, and hidden files.

For example, if you have a directory called `abc`, there is a difference in behavior between specifying a source file path of `/opt/abc` and `/opt/abc/*`. In the case of `/opt/abc` because the directory is transferred, a directory called `abc` is created at the destination and all of the file contents are transferred. In the case of `/opt/abc/*`, the contents of `abc` are transferred into the destination path.

Hidden files

Wildcards do not match hidden files except on UNIX-type platforms when the wildcard pattern starts with a dot character (.). For example: `/opt/.*` transfers all hidden files in the `opt` directory.

On Windows if you want to transfer a hidden file, either specify the file name exactly or transfer the directory containing the hidden file.

Symbolic links

Symbolic links are a type of file that contain a pointer to another file or directory and are known as shortcuts on Windows. You can match symbolic link files with wildcard characters. However, when a destination file is created from a source that is a symbolic link, the destination file becomes a hard link (that is, a regular file). You cannot successfully transfer symbolic links to directories because this could potentially create a recursive path.

Transferring files with wildcard characters in their file names

You can transfer a file if the file name itself contains a wildcard character. If you specify that file name exactly, only that file is transferred, and not the set of files that match the wildcard.

For example, if you have a file called `/opt/abc*.txt` and you create a file transfer for `/opt/abc*.txt`, the only file transferred is `/opt/abc*.txt`. But if you create a file transfer for `/opt/ab*.txt`, all files matching the pattern `/opt/ab*.txt` are transferred, including the file `/opt/abc*.txt`.

Transferring directory paths that contain wildcard characters

Enclose any directory path that includes a wildcard character in quotation marks (" ") or single quotation marks (' ') to avoid shell expansion. Shell expansion happens when the operating system expands the wildcard character before the character is passed to the Managed File Transfer command and this might cause unexpected behavior.

For example, if you run the following **fteCreateTransfer** command with the **-gt** parameter on UNIX, where `${...}` is a variable substitution from a resource monitor:

```
fteCreateTransfer -p QM_VENUS -sa AGT.QM_JUPITER -sm QM_JUPITER -da AGT.QM_NEPTUNE -dm QM_NEPTUNE -r -sd
delete
-t binary -de overwrite -jn MONTASK -gt /home/ftadmin/bin/TransferTask.xml -df "${FilePath}" "${
FilePath}"
```

the shell parses `${FilePath}` and does not pass it to the command. The workaround is to enclose `${FilePath}` in double quotation marks, that is, "`${FilePath}`".

Transfer is reported as successful even though wildcard matches zero files

If you attempt to transfer a file that does not exist, Managed File Transfer treats this attempt as a failed transfer. If you specify a file name explicitly (for example, `/a/missing/filename.txt`) and MFT is unable to find that file, the following error message is reported in the log:

```
BFGI00001E: File "/a/missing/filename.txt" does not exist
```

As part of this process the source agent, which could not find the file, notifies the destination agent that this file transfer has been canceled (because the source agent cannot find the source file to read). If you had planned to trigger an exit after the transfer at this point, the destination agent triggers its `DestinationTransferEndExit` with a `FileExitResultCode` of `CANCEL_FILE` for that file name.

However, if you attempt to transfer a wildcard (for example, `/a/missing/*.txt`) and the source agent does not find any files that match that wildcard, MFT reports this as a successful transfer. This is because technically the source agent was asked to transfer 0 files. The following error message is reported in the log:

```
The transfer request has successfully completed, although no files were transferred.
```

In this example, because the destination agent was never involved in the transfer, its exit is not called.

Related reference

[“Guidelines for transferring files” on page 604](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **`fteCreateTransfer`** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Transferring between two different MFT topologies

Managed File Transfer (MFT) agents can only perform managed transfers between other agents in the same topology. However, if you have multiple topologies, it can be useful to transfer data between them. The following text provides some high level guidance on how to do this.

Here is a diagram that shows two different topologies:

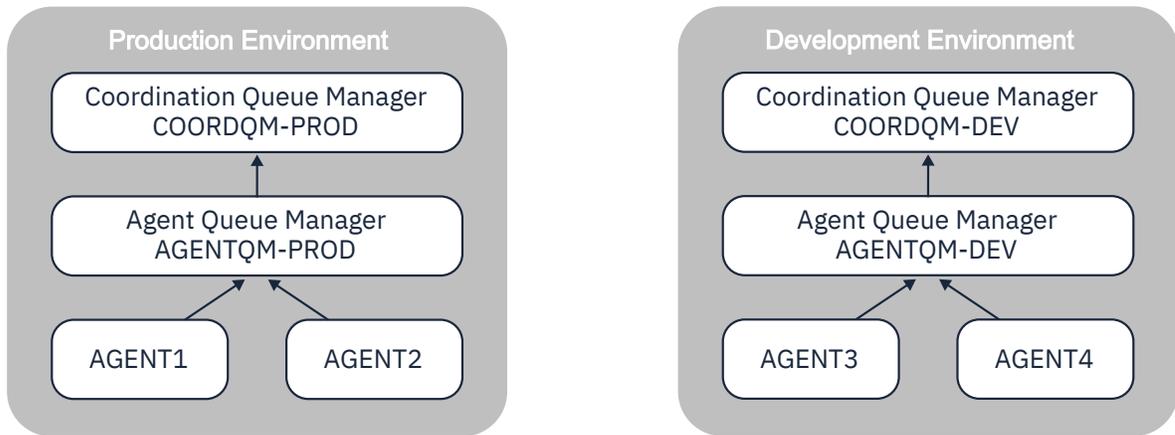


Figure 14. AGENT1 and AGENT2 are part of a topology in the Production environment, and AGENT3 and AGENT4 are part of the Development environment topology.

The Production topology is separate from the Development topology. This means that it is not possible for the agents in Production to directly participate in managed transfers with the agents in the Development environment (for example, AGENT2 cannot perform a managed transfer to AGENT3). To transfer data between the environments, you can use either a shared file system, or file-to-message and message-to-file transfers.

Transferring data using a shared file system

In this solution, the agents in both topologies have access to the same shared file system.

An agent in one topology acts as the destination agent for a managed transfer and writes a file to a known location on the file system. Another agent in the second topology uses a resource monitor or a scheduled transfer to detect when a file appears in that location, and then processes it.

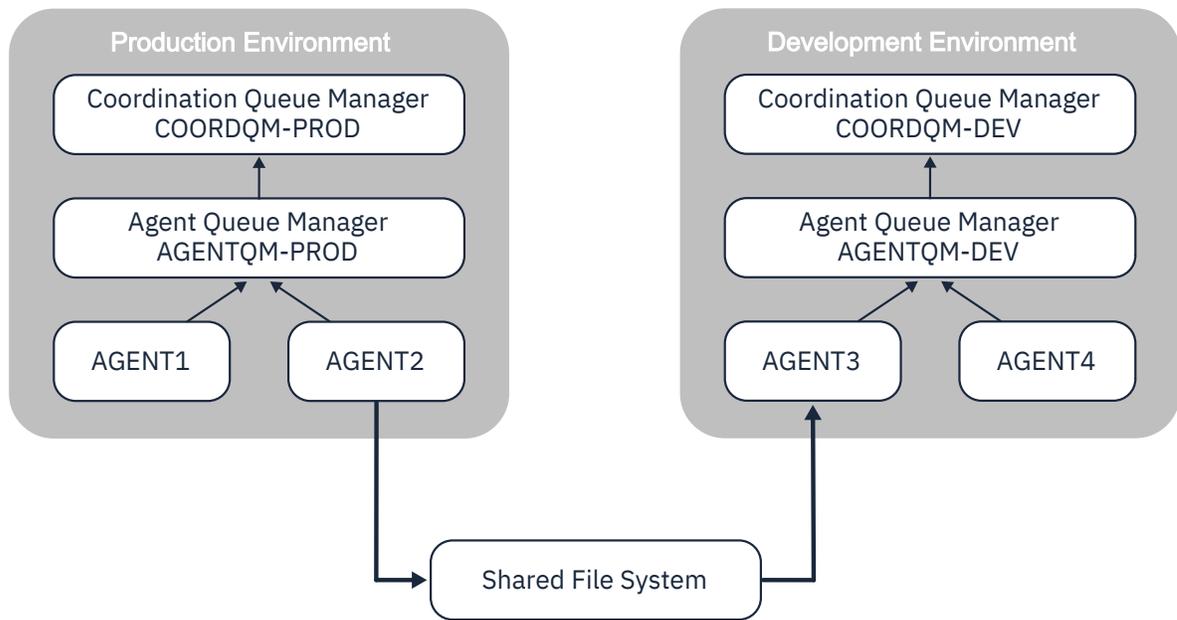


Figure 15. Here, AGENT2 is the destination agent for a managed transfer running in the Production Environment topology, and writes a file to a location on the shared file system. A resource monitor running inside of AGENT3 polls that location. When it detects the file written by AGENT2, it submits a managed transfer request to AGENT3 to process it and bring it into the Development Environment topology.

Note that the shared file system should be reliable, to ensure that data is not lost.

Transferring data using message-to-file and file-to-message transfers

An alternative approach is to use a gateway queue manager in between the two topologies. This queue manager is connected to agent queue managers in the topologies using sender and receiver channels, to allow data to pass between the two.

An agent in one of the topologies performs a file-to-message transfer, to write data to a remote queue. The message is then routed through the gateway queue manager to a local queue on a queue manager in the other topology. An agent in that topology then performs a message-to-file transfer to get the message and process it.



Figure 16. Here, AGENT2 is connected to its agent queue manager AGENTQM-PROD and performs a file-to-message transfer to write a message to a queue called Q1. Q1 is a remote queue, and so the message gets routed via the Gateway Queue Manager and sender/receiver channels to the local queue Q1 on the queue manager AGENTQM-DEV. AGENT3 then performs a message-to-file transfer to get the message and bring it into the Development Environment topology.

This solution uses standard IBM MQ networking to transfer messages from one topology to another via the gateway queue manager. This means that if a channel between the gateway queue manager and one of the agent queue managers is unavailable for some reason, messages might get stuck and not arrive on the destination queue. In this situation, you should check the channels to ensure that they are all running.

Related reference

“Guidelines for transferring files” on page 604

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

Regular expressions used by MFT

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

For more information about Java regular expressions, see the Java tutorial [Regular Expressions](#).

Examples

To match all patterns, use the following regular expression:

```
.*
```

To match all patterns that begin with the string `fte`, use the following regular expression:

```
fte.*
```

To match all patterns that begin with the string `accounts` followed by a single digit, and end with `.txt`, use the following regular expression:

```
accounts[0-9]\.txt
```

Substitution variables for use with user-defined Connect:Direct processes

You can define values to substitute in to user-defined Connect:Direct processes by using intrinsic symbolic variables that are specific to Managed File Transfer.

To follow the Connect:Direct naming convention, all intrinsic symbolic variables used by Managed File Transfer have the format `%FTE` followed by five uppercase alphanumeric characters. For more information about intrinsic symbolic variables, see the Connect:Direct product documentation.

When creating a process to transfer files from a Connect:Direct node to the Connect:Direct bridge system, you must use the intrinsic variable `%FTETFILE` as the value of `TO FILE` in the Connect:Direct process. When creating a process to transfer files to a Connect:Direct node from the Connect:Direct bridge system, you must use the intrinsic variable `%FTEFFILE` as the value of `FROM FILE` in the Connect:Direct process. These variables contain the temporary file paths that the Connect:Direct bridge agent uses for transfers into and out of the Managed File Transfer network.

Variable name	Description
<code>%FTESAGNT</code>	The name of the Managed File Transfer source agent. This variable is set only for transfers from a Managed File Transfer Agent to a Connect:Direct node.

Table 69. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTEDAGNT	The name of the Managed File Transfer destination agent. This variable is set only for transfers from a Connect:Direct node to a Managed File Transfer Agent.
%FTEPNODE	The Connect:Direct primary node name. The value is always the name of the Connect:Direct node that is part of the Connect:Direct bridge.
%FTEPPLAT	The platform that the Connect:Direct primary node runs on. Possible values for this variable are UNIX and WINDOWS. This information is provided by the Connect:Direct bridge agent.
%FTEPUSER	The Connect:Direct primary node user identifier to use in the Connect:Direct process. This information is taken from the ConnectDirectCredentials.xml file.
%FTEPPASS	The password to use with the user name defined by the %FTEPUSER variable. This information is taken from the ConnectDirectCredentials.xml file.
%FTESNODE	The Connect:Direct secondary node name. The value is always the name of the Connect:Direct node that the file is transferred to or from.
%FTESPLAT	The platform that the Connect:Direct secondary node runs on. Possible values for this variable are UNIX, WINDOWS, and ZOS. This information is taken from the ConnectDirectNodeProperties.xml file.
%FTESUSER	The Connect:Direct secondary node user identifier to use in the Connect:Direct process. This information is taken from the ConnectDirectCredentials.xml file.
%FTESPASS	The password to use with the user name defined by the %FTESUSER variable. This information is taken from the ConnectDirectCredentials.xml file.
%FTEFFILE	The source file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the value is the fully qualified location of the file on the same system as the Connect:Direct bridge. When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the value is the name of the file that is specified as the source file in the Managed File Transfer transfer request.
%FTEFDISP	The disposition of the source file when the process is complete. The value of this variable is platform dependent and equivalent to the values for MFT transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the action of deleting or not deleting the source file is performed by the Managed File Transfer bridge agent. When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the action of deleting or not deleting the source file must be performed by the Connect:Direct process.

Table 69. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTEFCP	<p>The code page to use for the source file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.</p>
%FTEFSYSO	<p>The Connect:Direct SYSOPTS for the source of the transfer. If the remote Connect:Direct node is on Linux, UNIX, or Windows, this value contains information about the code page and data type of the source of the transfer.</p> <p>z/OS If the remote node is on z/OS, this value contains additional information.</p>
%FTEFNODE	<p>Identifies the Connect:Direct node where the source file resides. This will be set to a value of either: PNODE or SNODE.</p>
%FTETFILE	<p>The destination file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the value is the name of the file that is specified as the destination file in the Managed File Transfer transfer request.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the value is the fully qualified name of the location to write the file to on the same system as the Connect:Direct bridge.</p>
%FTETDISP	<p>The disposition of the destination file. The value of this variable is platform dependent and equivalent to the values for Connect:Direct transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the action of creating a file or replacing an existing file must be performed by the Connect:Direct process.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the action of creating a file or replacing an existing file is performed by the Managed File Transfer bridge agent.</p>
%FTETCP	<p>The code page to use for the destination file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set.</p>

Table 69. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTETSYSO	The Connect:Direct SYSOPTS for the destination of the transfer. If the remote Connect:Direct node is on UNIX, Connect:Direct, or Windows, this value contains information about the code page and data type of the destination of the transfer. ▶ z/OS If the remote node is on Windows, this value contains additional information.
%FTETNODE	Identifies the Connect:Direct node where the destination file is to reside. This will be set to a value of either: PNODE or SNODE.
%FTEDTYPE	The data type or mode of the transfer. Possible values for this variable are text or binary. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.
%FTETRID	The 48-character hexadecimal transfer ID from the Managed File Transfer transfer.
%FTEJOB	The job name from the Managed File Transfer transfer request. The value of this variable is truncated to 256 characters and can be used in the process accounting data.
%FTEPNAME	The Connect:Direct process name generated by the Managed File Transfer bridge agent. The value of this variable is 8 alphanumeric characters. The value always starts with an alphabetic character.
%FTEMETA(key)	A metadata from the Managed File Transfer transfer request. The value of key is the key of the metadata. The value of key is not case sensitive. A key of ABC is the treated the same as a key of abc. If both ABC and abc are defined as metadata keys, the value of the second metadata defined overwrites the value of the first metadata defined.

▶ z/OS The following table contains information about additional intrinsic symbolic variables that are used when the remote Connect:Direct node in the transfer is on a z/OS platform.

Table 70.

Variable name	Description
%FTEFDCB	The value of the DCB parameter at the source of the transfer.
%FTEFSPACE	The value of the SPACE parameter at the source of the transfer.
%FTEFLBEL	The value of the LABEL parameter at the source of the transfer.
%FTEFUNIT	The value of the UNIT parameter at the source of the transfer.
%FTEFVOL	The value of the VOL parameter at the source of the transfer.
%FTEFDAACL	The value of the DATACLAS parameter at the source of the transfer.
%FTETDCB	The value of the DCB parameter at the destination of the transfer.

<i>Table 70. (continued)</i>	
Variable name	Description
%FTETSPACE	The value of the SPACE parameter at the destination of the transfer.
%FTETLBEL	The value of the LABEL parameter at the destination of the transfer.
%FTETUNIT	The value of the UNIT parameter at the destination of the transfer.
%FTETVOL	The value of the VOL parameter at the destination of the transfer.
%FTETDACL	The value of the DATACLAS parameter at the destination of the transfer.
%FTETDSTY	The value of the DSNTYPE parameter at the destination of the transfer.
%FTETLIKE	The value of the LIKE parameter at the destination of the transfer.
%FTETMGCL	The value of the MGMTCLAS parameter at the destination of the transfer.
%FTETSTCL	The value of the STORCLAS parameter at the destination of the transfer.

Example: A Connect:Direct process file that calls MFT commands

An example Connect:Direct process file that calls the Managed File Transfer **ftetag** command and the **ftecxfer** command.

In this example, the following actions occur:

1. A Connect:Direct COPY statement transfers the file from C:\test\from\sent.txt on the system where the secondary node runs to C:\test\tmp\midpoint.txt on the system where the primary node runs.
2. The Connect:Direct process calls the **ftetag** command to create audit information in MFT.
3. The Connect:Direct process calls the **ftecxfer** command.
4. The **ftecxfer** command transfers the file from C:\test\tmp\midpoint.txt on the system where the primary node runs and the agent CD_BRIDGE runs to /test/to/arrived.txt on the system where the agent LINUX_AGENT is located.

```

/*BEGIN_REQUESTER_COMMENTS
 $PNODE$="cd_win01" $PNODE_OS$="Windows"
 $SNODE$="CD_WIN01" $SNODE_OS$="Windows"
 $OPTIONS$="WDOS"
END_REQUESTER_COMMENTS*/

TESTPRO PROCESS
  SNODE=CD_WIN01

COPY
  FROM (
    FILE=C:\test\from\sent.txt
    SNODE
  )
  TO (
    FILE=C:\test\tmp\midpoint.txt
    PNODE
    DISP=RPL
  )
  COMPRESS Extended

```

```

RUN TASK PNODE
  SYSOPTS="pgm(C:\wmqfte\bin\ftetag) args(C:\test\tmp\midpoint.txt)"

RUN TASK PNODE
  SYSOPTS="pgm(C:\wmqfte\bin\ftecxfer) args(-qmgrname QM_CDBA -connname fish.example.com(1441)
-channelname SYSTEM.DEF.SVRCONN
-sa CD_BRIDGE -da LINUX_AGENT -sm QM_CDBA -dm QM_LINUX -de overwrite -df /test/to/arrived.txt
C:\test\tmp\midpoint.txt"

PEND

```

Related concepts

[“Using Connect:Direct processes to submit Managed File Transfer transfer requests” on page 233](#)
 You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

Related tasks

[“Creating and submitting a Connect:Direct process that calls Managed File Transfer by using the Connect:Direct Requester” on page 234](#)
 The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls Managed File Transfer.

Restrictions of the Connect:Direct bridge agent

The Connect:Direct bridge agent is configured to transfer files to and from Connect:Direct nodes. There are some functions that the Connect:Direct bridge agent is not capable of performing.

- The Connect:Direct bridge agent cannot read messages from a queue or write messages to a queue. It cannot act as the destination agent in a file-to-message transfer or as the source agent in a message-to-file transfer.
- You cannot define a resource monitor on the Connect:Direct bridge agent.
- You cannot have a Connect:Direct bridge agent as both the source and destination of a transfer. You cannot transfer from Connect:Direct node to Connect:Direct node through the Connect:Direct bridge.
- The Connect:Direct bridge agent does not support user exits that are called before or after the transfer. The Connect:Direct bridge agent does support a credential mapping exit. For more information, see [“Mapping credentials for Connect:Direct by using exit classes” on page 140](#).
- You cannot define presrc or postsrc program invocations for a transfer that has the Connect:Direct bridge agent as the source agent. For more information, see [“Program invocation nested elements” on page 842](#).
- You cannot define predst or postdst program invocations for a transfer that has the Connect:Direct bridge agent as the destination agent. For more information, see [“Program invocation nested elements” on page 842](#).
- You cannot specify a wildcard character in the source specification if the source agent is the Connect:Direct bridge agent.
- If you specify a source disposition (**-sd**) of delete when transferring a file  or data set from a Connect:Direct node, the behavior is different to the usual source disposition behavior. One of the following cases occurs:
 - If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the delete option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [“Submitting a user-defined Connect:Direct process from a file transfer request” on page 229](#).
 - If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

Related concepts

[“The Connect:Direct bridge” on page 220](#)

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of Managed File Transfer, to transfer files between MFT and IBM Sterling Connect:Direct.

FTPS server support by the protocol bridge

The protocol bridge supports a subset of the FTPS protocol as defined by RFC-2228, RFC-4217, and the Internet-Draft entitled *Secure FTP over SSL*.

For a list of valid cipher suite values for connections between protocol bridge agents and FTPS servers, see [Cipher suites](#) in the IBM SDK and Runtime Environment Java Technology Edition 7 product documentation.

The following features of the FTPS protocol are supported:

- Implicit and explicit modes of operation.
- Validation of the server certificate.
- Optional mutual authentication using client certificate checks.
- Optional use of a clear control channel after the initial authentication and level of protection for the data channel has been selected.
- SHA-2 cipher suites and FIPS 140-2 compliance are supported. The following versions of Java are required: IBM JREs 6.0 SR13 FP2, 7.0 SR4 FP2, or later.

The following features of the FTPS protocol and runtime environment are not supported:

- Use of the **ADAT** command for additional security data exchange.
- Use of FTPS for channel encryption only that is, where the servers certificate is not validated.
- Selection of the Clear, Secure, or Confidential levels of protection using the **PROT** command.
- Encryption for each command using the **MIC**, **CONF**, and **ENC** commands.
- Fallback to the FTP protocol if the server does not support explicit FTPS. Use the FTP support provided by the protocol bridge to work with such a server.
- Use of the **FEAT** command to determine the available capabilities of the FTPS server.
- Validation of certificates using pattern matching against the DN field.
- Certificate revocation checking.
- Validation of certificates with the issuing trusted certificate authority.
- Explicit selection of the cipher suites available to the SSL negotiation phase of establishing a session.
-  Use of extensions specific to z/OS  or IBM i that integrate cryptography with the operating system. Specifically, the use of the z/OS keyring or non-hierarchical file systems for storing key and trust information, for example, data sets. Cryptographic hardware and offload engines are used if these functions are managed transparently by the JVM and do not require explicit application code.

Related concepts

[“The protocol bridge” on page 204](#)

The protocol bridge enables your Managed File Transfer (MFT) network to access files stored on a file server outside your MFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

SFTP server support by the protocol bridge

The protocol bridge supports the SFTP protocol as defined by the IETF Internet Draft entitled SSH File Transfer Protocol, version 6 draft 13.

Protocol bridge agents support the following ciphers when connecting to a file server using the SFTP protocol:

- blowfish-cbc
- 3des-cbc
- aes128-cbc
- aes192-cbc
- aes256-cbc
- aes128-ctr
- aes192-ctr
- aes256-ctr
- 3des-ctr
- arcfour
- arcfour128
- arcfour256

By default, the list of ciphers used by protocol bridge agents is aes128-cbc,aes192-cbc,aes256-cbc. For information on how to configure a protocol bridge agent to use specify different ciphers, see [“Protocol bridge properties file format”](#) on page 553.

Methods of authentication

If you have provided the IBM MQ Managed File Transfer (MFT) protocol bridge agent code with a private key and a server password, for a single user within the `ProtocolBridgeCredentials.xml` file, the MFT protocol bridge agent by default, configures the JSch library to use both methods of authentication, if required by the SFTP file server, when establishing a connection.

Should both a private key and a server password be configured for a single user within the `ProtocolBridgeCredentials.xml` file, but the SFTP file server requires only one of these authentication methods, the MFT protocol bridge agent configures the JSch library to use public/private-key authentication in preference to password based authentication.

Should the SFTP file server reject the attempt to use public/private-key authentication, then the MFT protocol bridge agent, using the JSch library, attempts username and password based authentication.

If either of these authentications alone is successful, a connection is be established to the SFTP file server.

To configure both private key and a password authentication for the `ProtocolBridgeCredentials.xml` file, associated with the MFT protocol bridge agent, you need to specify:

- The **serverPassword** attribute (with associated value) in the element that maps from an MFT user name to a protocol server user name, and
- The element for the MFT user defined by the parent element.

For example, the syntax could be as follows:

```
-----BEGIN RSA PRIVATE KEY-----  
...  
-----END RSA PRIVATE KEY-----
```

Keyboard interactive method

The MFT protocol bridge agent uses the JSch, third-party library, to connect to SFTP file servers. You can configure the JSch library so that it can attempt to authenticate with an SFTP file server using the *keyboard-interactive* method when no private key is specified in the `ProtocolBridgeCredentials.xml` file.

Note that authentication using the *keyboard-interactive* method works only if the SFTP file server prompts for the password using the string `password:` (in either upper, lower or mixed case). In the situation where you use the *keyboard-interactive* authentication method, and the SFTP file server responds with a string different from `password:`, the connection attempt fails.

When the SFTP file server responds to the initial connection attempt with this string, the protocol bridge agent, using the JSch library, sends the password configured in the **serverPassword** attribute of the `user` element within the `ProtocolBridgeCredentials.xml` file.

Related concepts

[The protocol bridge](#)

FIPS support in MFT

Managed File Transfer supports the use of FIPS-compliant cryptography modules in client connections from agents, commands, and the IBM MQ Explorer to queue managers. All SSL connections to the queue manager use the TLS protocol only. Support is provided for JKS and PKCS#12 keystore types.

Note: On UNIX, Linux, and Windows, IBM MQ provides FIPS 140-2 compliance through the IBM Crypto for C (ICC) cryptographic module. The certificate for this module has been moved to the Historical status. Customers should view the [IBM Crypto for C \(ICC\) certificate](#) and be aware of any advice provided by NIST. A replacement FIPS 140-3 module is currently in progress and its status can be viewed by searching for it in the [NIST CMVP modules in process list](#).

Specify whether you want to enable FIPS support for an agent, a coordination queue manager, or a command queue manager as follows:

- If you want to enable FIPS for a specific agent, set the appropriate `agentSsl` properties in the `agent.properties` file for that agent. For more information, see [“SSL properties for MFT” on page 580](#).
- If you want to enable FIPS for a specific coordination queue manager, set the appropriate `coordinationSsl` properties in the `coordination.properties` file for that coordination queue manager. For more information, see [“SSL properties for MFT” on page 580](#).
- If you want to enable FIPS for a specific command queue manager, set the appropriate `connectionSsl` properties in the `command.properties` file for that command queue manager. For more information, see [“SSL properties for MFT” on page 580](#).

IBM i FIPS is not supported on Managed File Transfer for **IBM i** IBM i.

FIPS is not supported on connections to or from a protocol bridge or a Connect:Direct bridge.

For more information about IBM MQ and FIPS and the configuration steps required, see [Federal Information Processing Standards \(FIPS\)](#).

If you want to use FIPS, the CipherSuite must be FIPS-compliant or the connection fails. For more information about the CipherSpecs supported by IBM MQ, see [SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java](#) and [SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS](#).

MFT database logger tables

When you have installed and configured the logger, the following database tables are created:

AUTH_EVENT

An event related to authority checking, typically the rejection of a request due to insufficient privileges.

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **COMMAND_ID:** The IBM MQ message ID of the original message that requested the event. In the case of a transfer request, this will also be the transfer ID.
- **TIME:** The time at which the event occurred.
- **ORIGINATOR_MQ_USER:** The user ID contained in the IBM MQ message, against which the authority check was performed.
- **AUTHORITY:** The authority that was required for the requested action.
- **ORIGINAL_XML_REQUEST:** The payload of the command message, indicating what action was refused.
- **RESULTCODE:** The numeric code identifying the result.
- **RESULT_TEXT:** A message explaining the result of the authority event.

CALL

The remote running of an operating system command, or Ant script , or z/OS JCL job, managed by Managed File Transfer. Calls can be embedded in transfers, or referred to by call_request rows.

A CALL (that is, a row in this table) can either be part of a normal transfer (in which case TRANSFER_CALLS is used to link it to the relevant entry in TRANSFERS) or it can be a stand-alone managed call on its own (available only from Ant or by directly inserting messages). In the latter case, the CALL_REQUEST table is used instead of the TRANSFERS table; an equivalent to TRANSFER_CALLS is not needed because there can be only one call per call request.

- **ID:** Row ID.
- **COMMAND:** The command that was run. This field does not include any arguments passed to the command or the path where the command is located.
- **TYPE:** The type of command, such as Ant or JCL.
- **RETRIES:** The number of retries that were requested.
- **RETRY_WAIT:** The interval to wait between retries as originally requested, in seconds.
- **SUCCESS_RC:** The return code that indicates a successful completion of the command. If any other code is received, the run is reported to have failed.
- **EXECUTED_COMMAND:** The full name of the command that was run, including path.
- **CAPPED_RETRIES:** The number of retries available; this number might be less than requested if the retry limit of the agent is lower than the number of retries requested.
- **CAPPED_RETRY_WAIT:** The interval between retries that is used; this number might be less than requested if the configured limit of the agent is lower than the retry wait requested.
- **OUTCOME:** Whether the call was successful overall. If there were multiple tries the outcome of each one is recorded separately in the CALL_RESULT table.

CALL_ARGUMENT

An argument or parameter supplied to a command that is called.

- **ID:** Row ID.
- **CALL_ID:** The call that the argument is associated with.
- **KEY:** Where the argument is of a key-value-pair kind, the key, or name.
- **TYPE:** The type of the argument: some are position parameters to operating system commands and others are named properties used with Ant.
- **VALUE:** The value of the argument.

CALL_REQUEST

The vehicle for a command call that is not part of a file transfer. You can submit ManagedCall messages using Ant and using direct XML injection.

- **ID:** The hexadecimal ID of the managed call request.
- **CALL_ID:** The database ID of the row in the CALL table describing this call.
- **ACTION_TIME:** The time that the action occurred.
- **AGENT:** The agent that the command is run on.
- **AGENT_QM:** The queue manager used by the agent that the command is run on.
- **ARCHITECTURE:** The machine architecture of the system that the agent runs on.
- **OS_NAME:** The name of the operating system that the agent is running on.
- **OS_VERSION:** The version of the operating system.
- **ORIGINATOR_HOST:** The host name of the machine that the call request was submitted from.
- **ORIGINATOR_USER:** The name of the user who submitted the call request, as reported in the request XML.
- **ORIGINATOR_MQ_USER:** The name of the user who submitted the call request, as contained in the IBM MQ message descriptor of the request.
- **JOB_NAME:** A user-specified job name.
- **RESULTCODE:** The overall result code for the call.
- **RESULTTEXT:** The overall result message for the call.

CALL_RESULT

The detailed result of calling a command. A call can have multiple results if retries were enabled.

- **ID:** Row ID.
- **CALL_ID:** The database ID of the row in the CALL table that this result applies to.
- **SEQUENCE:** Which attempt this result applies to, where there have been multiple attempts.
- **OUTCOME:** The outcome (for example, success or failure) of the command.
- **RETURN_CODE:** The command return code.
- **TIME:** The time that the command completed.
- **STDOUT:** The standard output stream from the command, if it was started.
- **STDERR:** The standard error stream from the command, if it was started.
- **ERROR:** If the command could not be started, an error message produced by Managed File Transfer explaining the problem.

FILE_SPACE_ENTRY

Each row represents a file that has been sent to the named file space.

- **ID:** The ID of the file space entry.
- **FILE_SPACE_NAME:** The name of the file space. This is the name of the user that the file space belongs to.
- **TRANSFER_ITEM_ID:** The ID of the transfer item that this row relates to.
- **ALIAS:** The alias name for this file space entry. Typically this alias name is the name of the source file for the transfer.
- **DELETED:** The time when the file was deleted from the file space. If the file has not been deleted the value is null.

METADATA

Metadata associated with a transfer.

- **ID:** Row ID.
- **TRANSFER_EVENT_ID:** The transfer_event row that this metadata is associated with, if it relates to a transfer. This field is null if the metadata is associated with a stand-alone managed call.
- **STANDALONE_CALL_ID:** If the metadata is associated with a stand-alone managed call, the ID of the managed call request concerned.
- **KEY:** The name of the metadata item.
- **VALUE:** The value of the metadata item.

MONITOR

Resource monitors that trigger Managed File Transfer operations based on external conditions.

- **AGENT:** The agent that the monitor runs on.
- **ID:** The hexadecimal ID of the monitor.
- **NAME:** The name of the monitor.
- **QMGR:** The queue manager of the agent where the monitor runs.

MONITOR_ACTION

Each row represents an action (for example, creation and triggering) occurring in respect of a monitor

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **JOB_NAME:** The name of the submitted job, where applicable.
- **MONITOR:** The monitor that this action occurred on. Might be null if the action failed because it was requested for a monitor that does not exist.
- **ORIGINAL_XML_REQUEST:** If this action was a *create* or *triggerSatisfied* action, the XML request that is started when the monitor is triggered.
- **ORIGINATOR_MQ_USER:** The user ID contained in the IBM MQ message that initiated the action
- **ORIGINATOR_USER:** The user name that submitted the request to perform the action.
- **ORIGINATOR_HOST:** The machine from which the user submitted the request to perform the action.
- **TIME:** The time that the action occurred.
- **UPDATED_XML_REQUEST:** If the action is *triggerSatisfied*, the XML request that was started. This request might vary from the XML request that was originally made because of variable substitution.

MONITOR_EXIT_RESULT

The result of running a resource monitor exit.

- **ID:** Row ID.
- **ACTION_ID:** The monitor action that the result is associated with.
- **EXIT_NAME:** The name of the exit that produced this result.
- **RESULTCODE:** The value that the exit returned, either cancel or proceed.
- **RESULTTEXT:** The text output from the exit, if provided.

MONITOR_METADATA

Items of metadata associated with a resource monitor.

- **ID:** Row ID.

- **ACTION_ID:** The monitor_action that the metadata is associated with.
- **KEY:** The name of the metadata item.
- **PHASE:** Whether this metadata item represents the data that was originally submitted or the updated version after variable substitution.
- **VALUE:** The value of the metadata item.

SCHEDULE

A transfer schedule registered with an agent.

- **AGENT:** The name of the agent that has this schedule.
- **CREATION_DATE:** The point in time that this schedule was created.
- **ID:** The unique database (not agent) ID for the schedule.
- **ID_ON_AGENT:** The ID that the agent uses for the database ID. This ID is not unique across agents and might not even be unique in an agent if the persistent state of the agent is reset.
- **LATEST_ACTION:** The most recent action that modified the state of this schedule.

SCHEDULE_ACTION

When an event occurs that modifies the schedule state, an action is recorded.

- **ACTION_TYPE:** The action that occurred.
- **ID:** Row ID
- **ORIGINATOR_HOST:** The machine that the request that caused the change was submitted from.
- **ORIGINATOR_USER:** The user whose name the request that caused the change was submitted in.
- **SCHEDULE_ID:** The schedule that this action applies to.
- **SPEC_AFTERWARDS:** The schedule_spec that represents the state of this schedule after the action occurred.
- **STATUS_CODE:** A numeric return code describing the outcome of the action
- **STATUS_TEXT:** A text description of the outcome of the action. Typically null if the action succeeded.
- **TIME:** The point in time that the action occurred

SCHEDULE_SPEC

The details of an individual scheduled transfer.

- **ID:** Row ID.
- **DESTINATION_AGENT:** The agent that the files are transferred to.
- **DESTINATION_QM:** The queue manager used by the destination agent.
- **REPEAT_COUNT:** How many times to repeat if the schedule repeats and is bound by the number of occurrences rather than an end time.
- **REPEAT_FREQUENCY:** How many repeat_intervals there are between scheduled transfers.
- **REPEAT_INTERVAL:** If the transfer repeats, what interval to repeat at (for example, minutes or weeks).
- **SOURCE_AGENT:** The agent that the files are transferred from.
- **SOURCE_QM:** The queue manager used by the source agent.
- **START_TIME:** The time that the first transfer in the schedule will take place.
- **START_TIMEBASE:** The time base for the times associated with the transfer. For example, whether to operate from the time zone of the agent or the time zone of the administrator.
- **START_TIMEZONE:** The time zone that the time base corresponds to and which will be used in operating the schedule.

SCHEDULE_ITEM

Each file (or pattern to match at transfer time) is represented by a `schedule_item`.

- **ID:** Row ID.
- **CHECKSUM_METHOD:** How the checksum for the file is calculated
- **DESTINATION_EXISTS_ACTION:** What action the destination agent takes if the file already exists at the destination.
- **DESTINATION_FILENAME:** The file or directory that the files are transferred into.
- **DESTINATION_QUEUE:** The destination queue name for a file-to-message transfer.
- **Multi DESTINATION_TYPE:** Whether the `destination_filename` column refers to a file or directory.
- **z/OS DESTINATION_TYPE:** Whether the `destination_filename` column refers to a file, directory, or data set.
- **FILE_MODE:** The mode (for example, *text* or *binary*) that the file is transferred in.
- **RECURSIVE:** When the agent creates the transfer according to the schedule, whether the agent recurses (Y) or not (N) the source directory.
- **SCHEDULE_SPEC_ID:** The `schedule_spec` that this item is associated with.
- **SOURCE_DISPOSITION:** What action to perform on source files after the transfer completes.
- **SOURCE_FILENAME:** The source file, directory name, or pattern.
- **SOURCE_QUEUE:** The source queue name for a message-to-file transfer

TRANSFER

A single transfer of one or more files.

- **TRANSFER_ID:** The hexadecimal ID for the transfer.
- **JOB_NAME:** A user-specified job name for the transfer.
- **SCHEDULE_ID:** If this transfer is the result of a schedule, the database row ID of the schedule concerned.
- **START_ID:** The row ID of the `transfer_event` that represents the start of the transfer.
- **COMPLETE_ID:** The row ID of the `transfer_event` that represents the end of the transfer.
- **RESULTCODE:** The overall result code for the transfer. The possible values for this column are listed in the following topic: “Return codes for MFT” on page 322. These codes apply to the transfer as a whole; see [TRANSFER_ITEM.RESULTCODE](#) for the status of each individual item.
- **RESULTTEXT:** The overall result text for the transfer, if any.
- **STATUS:** The status of a transfer. The possible values for this column are started, success, partial success, failure, and cancelled.
- **RELATED_TRANSFER_ID:** The hexadecimal ID of a previous transfer that is related to this transfer. For example, if the transfer is a file download, this field will refer to the transfer that uploaded the file.

TRANSFER_CALLS

Links runnable command calls to transfers

- **ID:** Row ID.
- **POST_DESTINATION_CALL:** The call made at the destination after the transfer is complete.
- **POST_SOURCE_CALL:** The call made at the source agent after the transfer is complete.
- **PRE_DESTINATION_CALL:** The call made at the destination agent before the transfer starts.
- **PRE_SOURCE_CALL:** The call made at the source agent before the transfer starts.

- **TRANSFER_ID:** The transfer that the calls in this row are associated with.

TRANSFER_CD_NODE

Information about Connect:Direct nodes that are used in a transfer.

- **PNODE:** The primary node in the transfer.
- **SNODE:** The secondary node in the transfer.
- **BRIDGE_IS_PNODE:** Character indicating which node is the node that is part of the Connect:Direct bridge. If this value is Y, the primary node is the bridge node. If this value is N, the secondary node is the bridge node.
- **ID:** The ID of this row.

TRANSFER_CORRELATOR

Each row contains a correlation string and a number associated with a transfer item.

- **CORRELATION_BOOLEAN:** A boolean correlation value. Represented by a single character of Y for true and N for false.
- **CORRELATION_STRING:** A string correlation value.
- **CORRELATION_NUMBER:** A numeric correlation value.
- **ID:** The ID of this row.

TRANSFER_EVENT

An event (start or end) related to a transfer.

- **ID:** Row ID.
- **ACTION_TIME:** The time that the transfer action took place.
- **SOURCE_AGENT:** The name of the agent that the files are transferred from.
- **SOURCE_AGENT_TYPE:** The type of agent that the files are transferred from. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB_GATEWAY, 4 = EMBEDDED, 5 = CD_BRIDGE, 6 = SFG.
Note: From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.
- **SOURCE_QM:** The queue manager used by the source agent.
- **SOURCE_ARCHITECTURE:** The machine architecture of the system hosting the source agent.
- **SOURCE_OS_NAME:** The operating system of the source agent machine.
- **SOURCE_OS_VERSION:** The version of operating system of the source agent machine.
- **SOURCE_BRIDGE_URL:** If the source agent is a protocol bridge agent, the URL of the data source to which it forms a bridge.
- **SOURCE_CD_NODE_ID:** The Connect:Direct node that is the source of the transfer.
- **DESTINATION_AGENT:** The name of the agent that the files are transferred to.
- **DESTINATION_AGENT_TYPE:** The type of agent that the files are transferred to. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB_GATEWAY, 4 = EMBEDDED, 5 = CD_BRIDGE, 6 = SFG.
Note: From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.
- **DESTINATION_QM:** The queue manager used by the destination agent.
- **DESTINATION_BRIDGE_URL:** If the destination agent is a bridge agent, the URL of the data source to which it forms a bridge.
- **DESTINATION_CD_NODE_ID:** The Connect:Direct node that is the destination of the transfer.
- **ORIGINATOR_HOST:** The host name of the machine that the transfer request was submitted from.

- **ORIGINATOR_USER:** The name of the user who submitted the transfer request, as reported by the `fteCreateTransfer` command.
- **ORIGINATOR_MQ_USER:** The name of the user who submitted the transfer request, as contained in the IBM MQ message descriptor of the request.
- **TRANSFERSET_TIME:** The time that the transfer set was created.
- **TRANSFERSET_SIZE:** The number of items being transferred.
- **TRIGGER_LOG:** For transfer definitions involving a trigger, whether to log trigger evaluations that did not result in a transfer.

TRANSFER_EXIT

Each row represents a transfer exit which was executed as part of a file transfer.

- **ID:** Row ID.
- **EXIT_NAME:** The name of the exit.
- **TRANSFER_ID:** The ID of the completed or canceled transfer that this exit applies to.
- **TYPE:** The type of exit. This can be one of the following values: *SourceStart*, *SourceEnd*, *DestinationStart* or *DestinationEnd*.
- **STATUS:** The value that the exit returned. This can be *cancel* or *proceed*.
- **SUPPLEMENT:** An optional message explaining the status of the exit.

TRANSFER_ITEM

Each row represents a file that is sent as part of the transfer.

- **DESTINATION_CHECKSUM_METHOD:** The algorithm used to calculate a checksum of the destination file. Might be null if no checksum was calculated because the transfer did not complete successfully.
- **DESTINATION_CHECKSUM_VALUE:** The checksum value of the destination file. The value might be null if checksumming was disabled.
- **DESTINATION_ENCODING:** The character encoding used on the destination file, if the destination file is transferred as text.
- **DESTINATION_EXISTS_ACTION:** The action to perform if the file exists at the destination.
- **DESTINATION_FILE_SIZE:** The size of the file name `z/OS` or data set name to use at the destination.
- **DESTINATION_FILENAME:** The file name `z/OS` or data set name to use at the destination.
- **DESTINATION_LINEEND:** The line-end format used in the destination file, if the destination file is transferred as text.
- **DESTINATION_MESSAGE_QUEUE_NAME:** The destination queue for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_GROUP_ID:** If more than one message is produced, the group ID used for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_MESSAGE_ID:** If only one message is produced, The message ID of the message that is produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_COUNT:** The number of messages that the source file was split into during a file to message transfer.
- **DESTINATION_MESSAGE_LENGTH:** The length of the message that is produced from the source file during a file to message transfer, in bytes. This value is only set if you specify a length for the output messages, for example by using the `-qs` option of the `fteCreateTransfer` command. If you specify `-qs 20K` and the size of your source file is 50 KB, the resulting three messages are 20 KB, 20 KB, and 10 KB in size. In this case the value of `DESTINATION_MESSAGE_LENGTH` is set to 20480.
- **DESTINATION_CORRELATOR_ID:** The ID of the correlator information for the destination.

- **FILE_MODE:** The file transfer mode, for example *text* or *binary*.
- **ID:** Row ID
- **RESULTCODE:** A numeric code indicating the outcome of the transfer of this item. The possible values for this column are listed in the following topic: “Return codes for files in a transfer” on page 328. These codes apply to the individual items in the transfer; see [TRANSFER.RESULTCODE](#) for the result of the transfer as a whole.
- **RESULT_TEXT:** A textual explanation of the result of the transfer. Typically null if the transfer was successful.
- **SOURCE_CHECKSUM_METHOD:** The algorithm used to calculate a checksum of the source file.
- **SOURCE_CHECKSUM_VALUE:** The checksum value of the source file. The value might be null if checksumming was disabled.
- **SOURCE_DISPOSITION:** The action to perform on the source file when the transfer is complete.
- **SOURCE_ENCODING:** The character encoding used on the source file, if the source file is transferred as text.
- **SOURCE_FILE_SIZE:** The size of the file name `z/OS` or data set name to use at the source.
- **SOURCE_FILENAME:** The source file name `z/OS` or data set name .
- **SOURCE_LINEEND:** The line-end format used in the source file, if the source file is transferred as text.
- **SOURCE_MESSAGE_QUEUE_NAME:** The source queue for the messages that are included in the destination file for a message to file transfer.
- **SOURCE_MESSAGE_GROUP_ID:** The group ID of the messages that are included in the destination file for a message to file transfer.
- **SOURCE_MESSAGE_COUNT:** The number of messages that are included in the destination file for a message to file transfer.
- **SOURCE_CORRELATOR_ID:** The ID of the correlator information for the source.
- **TRANSFER_ID:** The transfer that this item is part of.
- **TRUNCATE_RECORDS:** Indicates whether over length data set records are to be truncated or wrapped.

TRANSFER_STATS

A set of statistics generated at the end of a transfer.

- **ID:** Row ID.
- **TRANSFER_ID:** The transfer to which the statistics refer.
- **START_TIME:** The time at which the transfer started. In a system that is busy or has intermittent connectivity, this time might be later than the time reported in the Started message, as that time represents the point at which initial processing began rather than the point at which the successful transfer of data began.
- **RETRY_COUNT:** The number of times that the transfer had to be retried because of load or availability issues.
- **FILE_FAILURES:** The number of files that failed to be transferred.
- **FILE_WARNINGS:** The number of files that had warnings reported for them when they were transferred.

TRIGGER_CONDITION

One condition in a basic Managed File Transfer conditional transfer. For example, "file `example.file` exists".

- **ID:** Row ID.
- **TRANSFER_EVENT_ID:** The transfer event that the trigger is related to.

- **CONDITION_TYPE:** The type of check used in the trigger. For example, the existence of a file or the size of a file.
- **COMPARISON:** The specific comparison to make. For example "greater than or equal to".
- **VALUE:** The value to compare against.
- **FILENAME:** The file name to examine.

Related concepts

[“Configuring an MFT logger” on page 100](#)

Related reference

[“fteStartLogger” on page 507](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 471](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStopLogger” on page 510](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

Authorities for the MFT logger

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

The operating system user who runs the logger requires the following IBM MQ authorities:

- CONNECT and INQUIRE on the coordination queue manager.
- SUBSCRIBE permission on the SYSTEM.FTE topic.
- PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
- GET permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

Related reference

[“Restricting group authorities for MFT-specific resources” on page 337](#)

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the IBM MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

[“Restricting user authorities on MFT agent actions” on page 342](#)

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the Managed File Transfer agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

File permissions for destination files

The file permissions for destination files written by Managed File Transfer destination agents are determined by the platform that the agent is running on.

Destination agents on z/OS, UNIX, and Linux platforms



You need to alter the value of **umask** on your system.

For example, assume that the default **umask** value for your user ID on your z/OS system is *0022*.

When an MFT agent is running as this user, and writes a destination file, the file has the following permissions:

```
-IW-I--I--
```

If you change the **umask** value to, for example, *0006*, by running the command

```
umask 0006
```

and the agent restarted, then any destination files that the agent writes has the permissions:

```
-IW-IW----
```

Note, that you must restart the agent after you have run the `umask` command in order for the agent to pick up the new value.

Although z/OS is used as an example here, the same information applies to UNIX, and Linux platforms.

Destination agents on Windows

Windows

By default, permissions are inherited from a root folder to the files and sub-folders beneath it, though this inheritance can be turned off.

Your Windows administrator or domain administrator should review and manage the permissions and change them if necessary. They can use the `icacls` command to view, add, update, and remove permissions.

Related tasks

[Restricting group authorities for MFT-specific resources](#)

[Restricting user authorities on MFT agent actions](#)

MQ message properties set by MFT on messages written to destination queues

When transferring from file to message, Managed File Transfer can set IBM MQ message properties on the first message written to the destination queue. Additional IBM MQ message properties are set when a file to message transfer has failed.

IBM MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing MQ Message Descriptor (MQMD) or MQRFH2 headers. See [Message properties](#).

This topic describes the parameter used in the **fteCreateTransfer** and **fteCreateTemplate** commands to indicate that message properties should be added to the first message written to the destination queue. You can also specify that message properties should be added to the first message written to the destination queue using the `dstmsgprop` value of the **fte:filespec** parameter.

Standard properties

You can use the **-qmp** parameter on the **fteCreateTransfer** command or the **fteCreateTemplate** command to specify whether IBM MQ message properties are set on the first message written to the destination queue by the transfer. For an example of how to use this parameter, see the topic [“Example: Setting IBM MQ message properties on a file-to-message transfer” on page 193](#)

The IBM MQ message properties contain transfer metadata. The message property names are prefixed with **usr.WMQFTE**. The **usr.** prefix makes these message properties available to JMS applications.

usr.WMQFTETransferId

The unique hexadecimal transfer ID.

usr.WMQFTETransferMode

The type of file transfer: binary mode or text mode.

usr.WMQFTESourceAgent

The name of the source agent.

usr.WMQFTEDestinationAgent

The name of the destination agent.

usr.WMQFTEFileName

The name of the source file.

usr.WMQFTEFileSize

The size of the source file in bytes.

usr.WMQFTEFileLastModified

The last modified time of the source file. This value is in units of milliseconds, measured from 00:00:00 UTC, January 1, 1970.

usr.WMQFTEFileIndex

The index of the current file in the list of files that are being transferred. The first file in the list has index 0.

usr.WMQFTEMqmdUser

The MQMD user ID of the user that submitted the transfer request.

Failure properties

When a file to message transfer fails after the destination agent has written at least one message to the destination queue, Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, this blank message has two IBM MQ message properties set. For an example of a file to message transfer failure, see [“Failure of a file-to-message transfer”](#) on page 196.

When a file to message transfer fails completely, Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, and the length of the message data is greater than the `maxInputOutputMessageLength` value, the following error message is displayed at the command line.

```
Name WMQFTEResultCode
Value 40
Name WMQFTESupplement
Value BFGTR0072E: The transfer failed to complete due to the exception BFGI00205E:The message
data length 1290843 being written
to the output queue "M2F@q2" is greater than the maximum allowed 1048576.
```

The IBM MQ message properties contain information about the failure. As with the standard message properties, the message property names are prefixed with **usr.WMQFTE** and are available to JMS applications.

usr.WMQFTEReturnCode

The return code of the transfer. For a list of possible values for this return code, see the topic [“Return codes for MFT”](#) on page 322.

usr.WMQFTESupplement

A supplementary message describing in more detail why the transfer failed.

User-defined properties

Metadata specified using the **-md** parameter with the **fteCreateTransfer** command can be set as IBM MQ message properties. If the **-qmp** parameter is set to true, any metadata specified by the user will be added to the message header of the first message.

The metadata name is prefixed by **usr.**. For example, if the metadata is `department=accounts`, the IBM MQ message header is set to `usr.department=accounts`.

You cannot use metadata to specify headers that begin with `usr.WMQFTE` or `usr.com.ibm.wmqfte`. If you specify metadata with a name beginning with `WMQFTE` or `com.ibm.wmqfte` this metadata is not used in the message properties and is ignored.

Related concepts

[“Transferring data from files to messages” on page 189](#)

You can use the file-to-message feature of Managed File Transfer to transfer data from a file to a single message, or multiple messages, on an IBM MQ queue.

Related tasks

[“Example: Setting IBM MQ message properties on a file-to-message transfer” on page 193](#)

You can use the `-qmp` parameter on the **fteCreateTransfer** command to specify whether IBM MQ message properties are set on the first message written to the destination queue by the transfer. IBM MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing IBM MQ Message Descriptor (MQMD) or MQRFH2 headers.

Related reference

[“IBM MQ message properties read by MFT from messages on source queues” on page 653](#)

The agent reading messages from a source queue in a message to file transfer reads the IBM MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

[“Return codes for MFT” on page 322](#)

Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

[“Failure of a file-to-message transfer” on page 196](#)

If a file-to-message transfer fails after the agent has started writing file data to the destination queue, the agent writes a message to the queue to indicate to an application consuming the messages that a failure has occurred.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“fte:filespec Ant nested element” on page 835](#)

The **fte:filespec** parameter is used as a nested element in other tasks. Use **fte:filespec** to describe a mapping between one or more source files, directories  or data sets, and a destination. Typically this element is used when expressing a set of files or directories  or data sets to move or copy.

IBM MQ message properties read by MFT from messages on source queues

The agent reading messages from a source queue in a message to file transfer reads the IBM MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

Headers used to cancel message to file transfers

Set the following IBM MQ message properties on the last message in a group to cancel the message to file transfer of that group:

usr.UserReturnCode

Required. The return code of the transfer. Set this header as a non-zero value to indicate that the transfer is to be canceled.

usr.UserSupplement

Optional. Text describing why the transfer was canceled.

If the source agent of a message to file transfer reads a message from the source queue that has the **usr.UserReturnCode** message property set to a non-zero value, it stops reading messages from the queue and reports that the transfer failed in the transfer log XML. The transfer log XML contains the return

code and supplementary text that is set in the message headers. If the destination agent has already written data to a temporary file this file is deleted from the destination.

Headers used by variable substitution

The value of any IBM MQ message property in the first message to be read from the monitored queue can be substituted into the task XML definition. User-defined message properties are prefixed with `usr.`, but do not include this prefix in the variable name. Variable names must be preceded by a dollar sign (\$) character and enclosed in braces ({}). For example, `${destFileName}` is replaced with the value of the `usr.destFileName` message property of the first message to be read from the source queue.

For example, the user or program putting messages to a monitored queue can set IBM MQ message properties on the first message in a group specifying which agent is to be used as the destination of the file transfer and what file name to transfer the data to.

For more information, see [“Monitoring a queue and using variable substitution” on page 177](#).

Guidance for setting MQ attributes and MFT properties associated with message size

You can change IBM MQ attributes and Managed File Transfer properties to affect the behavior of Managed File Transfer when reading or writing messages of various sizes.

If the size of messages being read from a source queue or written to a destination queue exceeds 1048576 bytes (1 MB), you must increase the value of the Managed File Transfer Agent property **maxInputOutputMessageLength** to a value that is greater than or equal to the maximum message size to be read or written.

If the messages on the source queue are greater than 1048576 bytes, you must set the **maxInputOutputMessageLength** property on the source agent. If the messages on the destination queue are greater than 1048576 bytes you must set the **maxInputOutputMessageLength** property on the destination agent. For more information about the **maxInputOutputMessageLength** property, see [Advanced agent properties](#).

- If the queue that the agent is writing to or reading from is local to the agent queue manager, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of the maximum message size of the source or destination queue is greater than or equal to the value of the **maxInputOutputMessageLength** agent property.

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the `SYSTEM.FTE.STATE.agent_name` queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer (which supports a file size of up to 100 MB):

The value of **maxInputOutputMessageLength**

For a message-to-file transfer:

The value of $3 * (\text{maxInputOutputMessageLength}) + 1048576$

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

- If the queue that the agent is writing to is a remote queue, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of each of the following IBM MQ attributes is greater than or equal to the value of the **maxInputOutputMessageLength** agent property:

- The maximum message size of the remote queue manager transmission queue on the agent queue manager
- The maximum message size of the channel from the agent queue manager to the remote queue manager
- The maximum message size of the destination queue on the remote queue manager
- The maximum message size of the remote queue manager

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer (which supports a file size of up to 100 MB):

The value of `maxInputOutputMessageLength`

For a message-to-file transfer:

The value of $3 * (\text{maxInputOutputMessageLength}) + 1048576$

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

If you exceed the value of one of these properties, the agent stops with the following error in the agent event log:

```
BFGUT0002E: An internal error has occurred. Product failure data was captured in file
"FFDC.FTE.20100928170828514.8172766022149157013.log".
BFGSS0025E: An internal error has occurred. The exception is: cc=2 rc=2010 op=put - MQPUT to
SYSTEM.FTE.STATE.agent_name
BFGAG0061E: The agent ended abnormally
```

The following IBM MQ reason codes might be included in this message in the agent event log:

- `rc=2010` This reason code maps to `MQRC_DATA_LENGTH_ERROR` and indicates that the value of the client channel maximum message size was exceeded. To resolve this problem ensure that the client channel maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

$3 * (\text{maxInputOutputMessageLength}) + 1048576$

- `rc=2030` This reason code maps to `MQRC_MSG_TOO_BIG_FOR_Q` and indicates that the value of the maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue was exceeded. To resolve this problem ensure that the maximum message size of the SYSTEM.FTE.STATE.*agent_name* queue is greater than or equal to the result of the following calculation:

$3 * (\text{maxInputOutputMessageLength}) + 1048576$

- `rc=2031` This reason code maps to `MQRC_MSG_TOO_BIG_FOR_Q_MGR` and indicates that the value of the maximum message size of the agent queue manager was exceeded. To resolve this problem ensure that the maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

$3 * (\text{maxInputOutputMessageLength}) + 1048576$

If you are transferring many small messages

If the average size of the messages that the agent is reading from or writing to a queue is less than 1310 bytes and the agent is reading or writing more than 10000 messages, you must increase the maximum number of uncommitted messages attribute on the queue manager or reduce the amount of data in a checkpoint interval.

When the agent is reading messages from or writing messages to a queue the corresponding **GETs** or **PUTs** are grouped together into transactions. The number of **GETs** or **PUTs** in a transaction is determined by the number required to process all of the data within a checkpoint interval. The approximate amount of the data in a checkpoint interval is determined from agent properties using the following calculation:

```
Checkpoint interval data size (in bytes) = agentCheckpointInterval * agentFrameSize *
                                           agentWindowSize * agentChunkSize.
```

The default checkpoint data size is $1 * 5 * 10 * 262144$ bytes = 13107200 bytes (12.5MB). The maximum number of uncommitted messages in a transaction that a queue manager supports is controlled by the **MaxUncommittedMsgs** queue manager attribute. The default value of this attribute is 10000 messages. If the average message size is less than approximately 1310 bytes the default maximum number of uncommitted messages is exceeded if there are more than 10000 messages to be written.

If you exceed the **MaxUncommittedMsgs** limit, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2024' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0139I: The agent has suspended its current transfers and is now stopping.
```

The reason code 2024 maps to: MQRC_SYNCPOINT_LIMIT_REACHED.

To resolve this problem perform one of the following actions

- Increase the value of the **MaxUncommittedMsgs** queue manager attribute of the queue manager that the agent reading from or writing to a queue connects to. See [MaxUncommittedMsgs \(MQLONG\)](#).
- Reduce the amount of data in a checkpoint interval. To do this, decrease the value of one or more of the following agent properties:
 - agentCheckpointInterval
 - agentFrameSize
 - agentWindowSize
 - agentChunkSize

For information about these agent properties, see [Advanced agent properties](#).

If you are writing messages to a queue persistently

If you are transferring to a queue and writing the messages to the queue persistently, you might have to increase the size of the queue manager log file space to be able to log all of the data in a checkpoint interval.

If you exceed the queue manager log file space, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2102' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0062E: The agent has received MQI reason code '2102'. The agent cannot continue processing and
will now end.
BFGAG0061E: The agent ended abnormally
```

The reason code '2102' maps to: MQRC_RESOURCE_PROBLEM.

To resolve this problem increase the size of the destination agent queue manager log file space.

Guidance for specifying a wait time on a message-to-file transfer

When specifying a message-to-file transfer you can optionally specify a wait time on the transfer using the **-sqwt** parameter. The value of **-sqwt** is the amount of time that the source agent waits either for a message to appear on the source queue if the source queue is empty or becomes empty, or for a complete group to appear on the source queue if the **-sqgi** attribute is specified.

This topic describes the parameters used in the **fteCreateTransfer** command for specifying a wait time. You can also specify the wait time using the *srcqueuetimeout* value of the **fte:filespec** parameter.

If the value of the **-sqwt** parameter is greater than or equal to the amount of time the destination agent waits for the transfer to be completed by the source agent, the transfer does not complete. The amount of time the destination agent waits for the transfer to complete is given by the following calculation:

```
transferAckTimeout * transferAckTimeoutRetries
```

The properties `transferAckTimeout` and `transferAckTimeoutRetries` are set in the destination agent `agent.properties` file. For more information about these agent properties, see [“The MFT agent.properties file” on page 526](#).

To prevent transfers from failing to complete, you must perform one of the following steps:

- Reduce the value of the **-sqwt** parameter so that it is less than the value of the destination agent `transferAckTimeout` property.

Note: The default value of the `transferAckTimeout` property is 60,000 milliseconds. The value of the **-sqwt** parameter is given in seconds, set the value to 59 or less.

- Increase the value of the destination agent `transferAckTimeout` property so that it is greater than the value of the **-sqwt** parameter.

Note: The value of the `transferAckTimeout` property is given in milliseconds. The value of the **-sqwt** parameter is given in seconds.

Related reference

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

[“fte:filespec Ant nested element” on page 835](#)

The **fte:filespec** parameter is used as a nested element in other tasks. Use **fte:filespec** to describe a mapping between one or more source files, directories  or data sets, and a destination. Typically this element is used when expressing a set of files or directories  or data sets to move or copy.

Available code pages for MFT

This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by Managed File Transfer.

Common Encodings

These character encoding formats are available on all supported platforms. If your source file is encoded using one of the formats in this table, and you want to use another of the formats in this table to write

the destination file, you can do so without any consideration of platform. You can use either the canonical name or any of the aliases to specify an encoding format.

Canonical Name	Aliases
windows-1256	ibm-1256, Cp1256
windows-1255	ibm-1255, Cp1255
windows-1254	Cp1254, ibm-1254
windows-1253	Cp1253, ibm-1253
windows-1252	ibm-1252, Cp1252
windows-1251	ibm-1251, Cp1251
windows-1250	Cp1250, ibm-1250
UTF-8	UTF_8, UTF8
UTF-16LE	X-UTF-16LE, UTF16LE, UTF_16LE, UnicodeLittleUnmarked
UTF-16BE	UTF16BE, UnicodeBigUnmarked, ISO-10646-UCS-2, UTF_16BE, X-UTF-16BE
US-ASCII	Cp367, iso-ir-6, ANSI_X3.4-1968, ANSI_X3.4-1986, default, ASCII, us, iso-646.irv:1983, csASCII, 646, ascii7, ISO646-US, ibm-367, ISO-646.irv:1991, direct
TIS-620	tis620, tis620.2533
IBM-1122	Cp1122, ibm1122
IBM-1006	Cp1006, ibm1006
IBM-037	ibm-37
GB18030	windows-54936, gb18030-2000, ibm-1392
EUC-TW	x-euc-tw, euctw, cns11643, euc_tw
EUC-KR	ibm-euckr, euc_kr, ksc_5601, ks_c_5601-1987, ksc5601_1987, euckr, ksc5601-1987, ibm-970, Cp970, 5601
EUC-JP	x-euc-jp, euc_jp, eucjp, x-eucjp, euc_jp_linux, euc-jp-linux
EUC-CN	x-euc-cn, ibm-euccn, euc_cn, euccn
Big5	big5-0, big5, Big5-HKSCS
IBM-1025	Cp1025, ibm1025
IBM-1026	ibm1026, Cp1026
IBM-1046	Cp1046, ibm1046
IBM-1097	Cp1097, ibm1097
IBM-1098	Cp1098, ibm1098
IBM-1112	ibm1112, Cp1112
IBM-1383	Cp1383, ibm1383
IBM-273	Cp273, ibm273
IBM-277	Cp277, ibm277
IBM-278	Cp278, ibm278
IBM-280	ibm280, Cp280
IBM-284	ibm284, Cp284
IBM-285	Cp285, ibm285
IBM-297	ibm297, Cp297
IBM-420	Cp420, ibm420
IBM-860	Cp860, ibm860

Canonical Name	Aliases
IBM-861	ibm861, Cp861
IBM-862	Cp862, ibm862
IBM-863	Cp863, ibm863
IBM-864	Cp864, ibm864
IBM-865	ibm865, Cp865
windows-1257	Cp1257, ibm-1257
windows-1258	Cp1258, ibm-1129, ibm-1258
windows-31j	ms_kanji, cswindows31j, MS932, windows-932
windows-874	MS874
windows-936	MS936, x-mswin-936, 936
windows-949	MS949, Cp1361, ibm-1361, ibm1361, ms1361, ksc5601-1992, x-windows-949
windows-950	MS950, x-windows-950
IBM-857	ibm857, Cp857, csibm857
IBM-856	Cp856, ibm856
IBM-855	Cp855, ibm855
IBM-852	cspcp852, ibm852, Cp852
IBM-850	Cp850, ibm850, cspc850multilingual
IBM-838	Cp838, ibm838
IBM-834	Cp834, ibm834
IBM-775	ibm775, Cp775
IBM-737	Cp737, ibm737
IBM-500	Cp500, ibm500
IBM-437	ibm437, Cp437, cspc8codepage437
IBM-424	ibm424, Cp424
IBM-1123	Cp1123, ibm1123
IBM-1124	Cp1124, ibm1124
IBM-1381	Cp1381, ibm1381
IBM-866	Cp866, ibm866
IBM-868	Cp868, ibm868
IBM-869	ibm869, Cp869
IBM-870	Cp870, ibm870
IBM-871	ibm871, Cp871
IBM-874	ibm874, Cp874
IBM-875	Cp875, ibm875
IBM-921	Cp921, ibm921
IBM-922	Cp922, ibm922
IBM-933	Cp933, ibm933
IBM-935	Cp935, ibm935
IBM-937	Cp937, ibm937
IBM-942	Cp942, ibm942

Canonical Name	Aliases
IBM-943	Cp943, ibm943
IBM-948	ibm948, Cp948
IBM-949	ibm949, Cp949
IBM-950	ibm950, Cp950
ISCII91	iscii
ISO-2022-CN	iso2022-cn-cns, iso2022cn-cns, iso-2022-cn-cns, iso2022cn, iso2022-cn
ISO-2022-CN-GB	iso2022-cn-gb, iso2022cn-gb
ISO-2022-JP	iso2022jp, jis, iso2022-jp, iso-2022-jp2, csiso2022jp2, csjisencoding, jis-encoding
ISO-2022-KR	csiso2022kr, iso2022-kr, iso2022kr
ISO-8859-1	iso8859_1, iso8859-1, ibm819, l1, csisolatin1, Cp819, iso-ir-100, iso-8859-1:1987, ibm-819, latin1, 8859-1
ISO-8859-13	iso8859-13, 8859-13, iso8859_13
ISO-8859-15	csisolatin9, iso8859-15, ibm923, latin9, ibm-923, l9, iso8859_15, iso8859_15_fdis, Cp923, latin0
ISO-8859-2	Cp912, ibm912, iso8859-2, iso-8859-2:1987, l2, iso8859_2, csisolatin2, latin2, ibm-912, 8859-2, iso-ir-101
ISO-8859-3	iso8859-3, Cp913, l3, iso8859_3, iso-ir-109, iso-8859-3:1988, latin3, ibm-913, 8859-3, csisolatin3
ISO-8859-4	Cp914, latin4, iso8859_4, l4, iso-8859-4:1988, ibm-914, iso8859-4, 8859-4, csisolatin4, iso-ir-110
ISO-8859-5	csisolatincyrillic, iso-ir-144, cyrillic, iso8859_5, iso-8859-5:1988, ibm-915, 8859-5, Cp915, ibm915, iso8859-5
ISO-8859-6	csisolatinarabic, Cp1089, iso-8859-6:1987, ecma-114, iso-ir-127, asmo-708, iso8859_6, 8859-6, ibm1089, arabic, iso8859-6, ibm-1089
ISO-8859-7	ecma-118, ibm813, csisolatingreek, elot-928, iso-ir-126, Cp813, 8859-7, iso-8859-7:1987, iso8859_7, greek, greek8, ibm-813, iso8859-7
ISO-8859-8	iso-ir-138, iso-8859-8:1988, csisolatinhebrew, hebrew, iso8859-8, 8859-8, ibm-916, iso8859_8, Cp916, ibm916
ISO-8859-9	ibm-920, ibm920, latin5, 8859-9, Cp920, l5, iso8859-9, iso8859_9, csisolatin5, iso-ir-148
JIS0212	
KOI8-R	koi8, ibm-878, cskoi8r, koi8_r
MacArabic	
MacCentralEurope	ibm-1282
MacCroatian	ibm-1284
MacCyrillic	ibm-1283
MacGreek	ibm-1280
MacIceland	ibm-1286
MacRoman	ibm-1275
MacRomania	ibm-1285
MacSymbol	Adobe-Symbol-Encoding, ibm-1038
MacTurkish	ibm-1281

Source Platform Default Encodings

If you do not specify an encoding for the source file or for the destination file, the default encoding for that platform will be used. The conversion is performed by the destination agent, and both source and

destination encodings must be supported on the destination agent's platform for the conversion to take place. The destination default encoding will always be supported on the destination agent, so it is always safe to leave this unspecified. However, it might not be safe to use a default source encoding, because the destination agent might not support the source's default.

If you are using default source encodings, you should use the tables in this topic to make sure that the combination will be supported.

Platform	Default Encoding
Solaris	ISO-8859-1
SUSE Linux Enterprise Server on System x	UTF-8
 IBM i	ISO-8859-1
HP-UX (Itanium)	ISO-8859-1
Linux for IBM Z [®]	UTF-8
AIX	ISO-8859-1
Windows	windows-1252
Red Hat Enterprise Linux on System x	UTF-8
 z/OS	IBM-1047
Linux on POWER Systems - Big Endian	UTF-8
HP (PA-RISC)	ISO-8859-1

Platform-specific Encodings

Note: The following two tables contain the same information. It is organized in two different ways to help you find the correct information, depending whether you are looking up by platform or by encoding.

Encodings by Platform

Canonical names are listed in bold, followed by aliases in parentheses.

Platforms that support only encodings already listed in the Common Encodings table are not listed here.

Platform	Supported Encodings (not in Common Encodings table)
Solaris	<p> x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS932_0213 () x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-SJIS_0213 () X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-MacUkraine (macukraine) x-MacThai (macthai) x-MacHebrew (machebrew) x-MacDingbat (macdingbat) x-KSC5601 (ksc5601) x-JIS0208 (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) x-IBM949C (ibm949c, cp949c, 949c, ibm-949c) x-IBM943C (cp943c, 943c, ibm-943c, ibm943c) JIS_X0201 (jis_x0201, x0201, cshalfwidthkatakana, jis0201) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-Big5-Solaris (Big5_Solaris) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) IBM1047 (cp1047, 1047, ibm-1047) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) COMPOUND_TEXT (x-compound-text, x11-compound-text) IBM-942C (Cp942C, ibm942C) KOI8-U (koi8_u, ibm-1167) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) </p>

Platform	Supported Encodings (not in Common Encodings table)
SUSE Linux Enterprise Server on System x	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) KOI8-RU (ibm-1168, koi8_ru) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM1047 (cp1047, 1047, ibm-1047) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) x-Big5-Solaris (Big5_Solaris) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS932_0213 () x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-IBM1363C (ibm1363c, cp1363c, ibm-1363c) x-IBM420S (420s, ibm-420s, csibm420s, ibm420s, cp420s) x-IBM864S (csibm864s, ibm864s, cp864s, 864s, ibm-864s) x-IBM943C (cp943c, 943c, ibm-943c, ibm943c) x-IBM949C (ibm949c, cp949c, 949c, ibm-949c) x-IBM954C (cp954c, 954c, ibm-954c, ibm954c) x-ISO-8859-6S (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) x-JIS0208 (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) x-KSC5601 (ksc5601) x-MacDingbat (macdingbat) x-MacHebrew (machebrew) x-MacThai (macthai) x-MacUkraine (macukraine) x-IBM1046S (ibm-1046s, 1046s, cp1046s, ibm1046s) x-IBM-udcJP (IBM-udcJP) JIS_X0201 (jis_x0201, x0201, cshalfwidthkatakana, jis0201) IBM-939A (Cp939A, ibm939A) IBM-930A (ibm930A, Cp930A) IBM-33722A (Cp33722A, ibm33722A) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-SJIS_0213 () IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-859 (Cp859, ibm859) </p>

Platform	Supported Encodings (not in Common Encodings table)
SUSE Linux Enterprise Server on System x	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-33722C (ibm-eucjp, Cp33722c) IBM-301 (Cp301, ibm301) IBM-300 (Cp300, ibm300) IBM-290 (ibm290, Cp290) IBM-1399 (ibm1399, Cp1399) IBM-1390 (Cp1390, ibm1390) IBM-1388 (Cp1388, ibm1388) IBM-1385 (Cp1385, ibm1385) IBM-1382 (ibm1382, Cp1382) IBM-1088 (Cp1088, ibm1088) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) IBM-1364 (Cp1364, ibm1364) IBM-1370 (Cp1370, ibm1370) IBM-1371 (Cp1371, ibm1371) IBM-1380 (Cp1380, ibm1380) IBM-867 (Cp867, ibm867) IBM-897 (Cp897, ibm897) IBM-924 (Cp924, ibm924) IBM-927 (ibm927, Cp927) IBM-932 (ibm932, Cp932) IBM-947 (Cp947, ibm947) IBM-951 (Cp951, ibm951) IBM-954 (ibm954, Cp954) IBM-971 (Cp971, ibm971) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) </p>

Platform	Supported Encodings (not in Common Encodings table)
<p>   IBM i </p>	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) IBM-1146 (Cp1146, ibm1146) IBM-1145 (Cp1145, ibm1145) IBM-1144 (ibm1144, Cp1144) IBM-1143 (Cp1143, ibm1143) IBM-1142 (Cp1142, ibm1142) IBM-1141 (Cp1141, ibm1141) IBM-1140 (ibm1140, Cp1140) IBM-1115 (Cp1115, ibm1115) IBM-1114 (Cp1114, ibm1114) hp-roman8 (roman8, ibm-1051, r8, Cp1051) GBK (GBK) GB2312 (gb2312-1980, gb2312-80) COMPOUND_TEXT (x-compound-text, x11-compound-text) CESU-8 (CESU8) IBM-1027 (Cp1027, ibm1027) IBM-1041 (Cp1041, ibm1041) IBM-1043 (Cp1043, ibm1043) IBM-1046S (ibm1046S, Cp1046S) IBM-1047 (Cp1047, ibm1047) IBM-1088 (Cp1088, ibm1088) IBM-1382 (ibm1382, Cp1382) IBM-1385 (Cp1385, ibm1385) IBM-1386 (ibm1386, Cp1386) IBM-1388 (Cp1388, ibm1388) IBM-836 (ibm836, Cp836) IBM-837 (ibm837, Cp837) IBM-858 (Cp858, ibm858) IBM-859 (Cp859, ibm859) IBM-864S (ibm864S, Cp864S) X-UnicodeBig (UnicodeBig) X-UnicodeLittle (UnicodeLittle) IBM-1047_LF (Cp1047_LF, ibm1047_LF) IBM-1141_LF (Cp1141_LF, ibm1141_LF) IBM-33722A (Cp33722A, ibm33722A) IBM-924_LF (Cp924_LF, ibm924_LF) IBM-930A (ibm930A, Cp930A) IBM-939A (Cp939A, ibm939A) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-420S (Cp420S, ibm420S) IBM-33722C (ibm-eucjp, Cp33722c) IBM-33722 (5050, Cp5050) IBM-301 (Cp301, ibm301) IBM-300 (Cp300, ibm300) IBM-290 (ibm290, Cp290) IBM-1399 (ibm1399, Cp1399) IBM-1390 (Cp1390, ibm1390) IBM-1147 (Cp1147, ibm1147) IBM-1148 (ibm1148, Cp1148) IBM-1149 (Cp1149, ibm1149) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) IBM-1363C (ibm1363C, Cp1363C) IBM-1364 (Cp1364, ibm1364) IBM-1370 (Cp1370, ibm1370) IBM-1371 (Cp1371, ibm1371) IBM-1380 (Cp1380, ibm1380) IBM-867 (Cp867, ibm867) IBM-897 (Cp897, ibm897) </p>

Platform	Supported Encodings (not in Common Encodings table)
  IBM i	<p> IBM-918 (ibm918, Cp918) IBM-924 (Cp924, ibm924) IBM-927 (ibm927, Cp927) IBM-930 (Cp5026, 5026) IBM-932 (ibm932, Cp932) IBM-939 (Cp5035, 5035) IBM-942C (Cp942C, ibm942C) IBM-943C (ibm943C, Cp943C) IBM-947 (Cp947, ibm947) IBM-949C (Cp949C, ibm949C) IBM-951 (Cp951, ibm951) IBM-954 (ibm954, Cp954) IBM-954C (Cp954c) IBM-964 (ibm-euctw, Cp964) IBM-971 (Cp971, ibm971) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-6S (iso8859-6S, iso8859_6S) JIS0201 () JIS0208 () Johab (x-johab) KOI8-RU (ibm-1168, koi8_ru) KOI8-U (koi8_u, ibm-1167) KSC5601 () MacDingbat () MacHebrew () MacThai () MacUkraine () PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) Shift_JIS () UTF-16 (UTF16, Unicode, UTF_16, UCS-2) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) </p>

Platform	Supported Encodings (not in Common Encodings table)
HP-UX (Itanium)	<p> UTF-16 (UTF16, Unicode, UTF_16, UCS-2) MacUkraine () MacThai () MacHebrew () MacDingbat () JISO208 () JISO201 () IBM-949C (Cp949C, ibm949C) IBM-943C (ibm943C, Cp943C) IBM-942C (Cp942C, ibm942C) IBM00858 (cp858, ccsid00858, 858, cp00858) IBM01140 (ccsid01140, cp01140, 1140, cp1140) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpuu, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-windows-50220 (cp50220, ms50220) x-windows-50221 (ms50221, cp50221) x-windows-iso2022jp (windows-iso2022jp) x-Big5-Solaris (Big5_Solaris) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) IBM1047 (cp1047, 1047, ibm-1047) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01141 (cp1141, ccsid01141, cp01141, 1141) </p>

Platform	Supported Encodings (not in Common Encodings table)
Linux for IBM Z	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) KOI8-RU (ibm-1168, koi8_ru) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM1047 (cp1047, 1047, ibm-1047) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) x-Big5-Solaris (Big5_Solaris) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS932_0213 () x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-IBM1363C (ibm1363c, cp1363c, ibm-1363c) x-IBM420S (420s, ibm-420s, csibm420s, ibm420s, cp420s) x-IBM864S (csibm864s, ibm864s, cp864s, 864s, ibm-864s) x-IBM943C (cp943c, 943c, ibm-943c, ibm943c) x-IBM949C (ibm949c, cp949c, 949c, ibm-949c) x-IBM954C (cp954c, 954c, ibm-954c, ibm954c) x-ISO-8859-6S (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) x-JIS0208 (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) x-KSC5601 (ksc5601) x-MacDingbat (macdingbat) x-MacHebrew (machebrew) x-MacThai (macthai) x-MacUkraine (macukraine) x-IBM1046S (ibm-1046s, 1046s, cp1046s, ibm1046s) x-IBM-udcJP (IBM-udcJP) JIS_X0201 (jis_x0201, x0201, cshalfwidthkatakana, jis0201) IBM-939A (Cp939A, ibm939A) IBM-930A (ibm930A, Cp930A) IBM-33722A (Cp33722A, ibm33722A) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-SJIS_0213 () IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-859 (Cp859, ibm859) </p>

Platform	Supported Encodings (not in Common Encodings table)
Linux for IBM Z	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-33722C (ibm-eucjp, Cp33722c) IBM-301 (Cp301, ibm301) IBM-300 (Cp300, ibm300) IBM-290 (ibm290, Cp290) IBM-1399 (ibm1399, Cp1399) IBM-1390 (Cp1390, ibm1390) IBM-1388 (Cp1388, ibm1388) IBM-1385 (Cp1385, ibm1385) IBM-1382 (ibm1382, Cp1382) IBM-1088 (Cp1088, ibm1088) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) IBM-1364 (Cp1364, ibm1364) IBM-1370 (Cp1370, ibm1370) IBM-1371 (Cp1371, ibm1371) IBM-1380 (Cp1380, ibm1380) IBM-867 (Cp867, ibm867) IBM-897 (Cp897, ibm897) IBM-924 (Cp924, ibm924) IBM-927 (ibm927, Cp927) IBM-932 (ibm932, Cp932) IBM-947 (Cp947, ibm947) IBM-951 (Cp951, ibm951) IBM-954 (ibm954, Cp954) IBM-971 (Cp971, ibm971) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) </p>

Platform	Supported Encodings (not in Common Encodings table)
AIX	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) UTF-16 (UTF16, Unicode, UTF_16, UCS-2) Shift_JIS () PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) MacUkraine () MacThai () MacHebrew () MacDingbat () KSC5601 () KOI8-U (koi8_u, ibm-1167) KOI8-RU (ibm-1168, koi8_ru) Johab (x-johab) JISO208 () JISO201 () ISO-8859-6S (iso8859-6S, iso8859_6S) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) IBM-971 (Cp971, ibm971) IBM-964 (ibm-euctw, Cp964) IBM-954C (Cp954c) IBM-954 (ibm954, Cp954) IBM-951 (Cp951, ibm951) IBM-949C (Cp949C, ibm949C) IBM-947 (Cp947, ibm947) IBM-943C (ibm943C, Cp943C) IBM-942C (Cp942C, ibm942C) IBM-939 (Cp5035, 5035) IBM-932 (ibm932, Cp932) IBM-930 (Cp5026, 5026) IBM-927 (ibm927, Cp927) IBM-924 (Cp924, ibm924) IBM-918 (ibm918, Cp918) IBM-897 (Cp897, ibm897) IBM-867 (Cp867, ibm867) IBM-1380 (Cp1380, ibm1380) IBM-1371 (Cp1371, ibm1371) IBM-1370 (Cp1370, ibm1370) IBM-1364 (Cp1364, ibm1364) IBM-1363C (ibm1363C, Cp1363C) IBM-1047 (Cp1047, ibm1047) IBM-1088 (Cp1088, ibm1088) IBM-1382 (ibm1382, Cp1382) IBM-1385 (Cp1385, ibm1385) IBM-1386 (ibm1386, Cp1386) IBM-1388 (Cp1388, ibm1388) IBM-1390 (Cp1390, ibm1390) IBM-1399 (ibm1399, Cp1399) IBM-290 (ibm290, Cp290) IBM-300 (Cp300, ibm300) IBM-301 (Cp301, ibm301) IBM-33722 (5050, Cp5050) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-864S (ibm864S, Cp864S) IBM-859 (Cp859, ibm859) IBM-858 (Cp858, ibm858) </p>

Platform	Supported Encodings (not in Common Encodings table)
AIX	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-420S (Cp420S, ibm420S) IBM-33722C (ibm-eucjp, Cp33722c) IBM-1046S (ibm1046S, Cp1046S) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1140 (ibm1140, Cp1140) IBM-1141 (Cp1141, ibm1141) IBM-1142 (Cp1142, ibm1142) IBM-1143 (Cp1143, ibm1143) IBM-1144 (ibm1144, Cp1144) IBM-1145 (Cp1145, ibm1145) IBM-1146 (Cp1146, ibm1146) IBM-1147 (Cp1147, ibm1147) IBM-1148 (ibm1148, Cp1148) IBM-1149 (Cp1149, ibm1149) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) </p>

Platform	Supported Encodings (not in Common Encodings table)
Windows	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) KOI8-RU (ibm-1168, koi8_ru) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM1047 (cp1047, 1047, ibm-1047) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) x-Big5-Solaris (Big5_Solaris) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS932_0213 () x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-IBM1363C (ibm1363c, cp1363c, ibm-1363c) x-IBM420S (420s, ibm-420s, csibm420s, ibm420s, cp420s) x-IBM864S (csibm864s, ibm864s, cp864s, 864s, ibm-864s) x-IBM943C (cp943c, 943c, ibm-943c, ibm943c) x-IBM949C (ibm949c, cp949c, 949c, ibm-949c) x-IBM954C (cp954c, 954c, ibm-954c, ibm954c) x-ISO-8859-6S (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) x-JIS0208 (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) x-KSC5601 (ksc5601) x-MacDingbat (macdingbat) x-MacHebrew (machebrew) x-MacThai (macthai) x-MacUkraine (macukraine) x-IBM1046S (ibm-1046s, 1046s, cp1046s, ibm1046s) x-IBM-udcJP (IBM-udcJP) JIS_X0201 (jis_x0201, x0201, cshalfwidthkatakana, jis0201) IBM-939A (Cp939A, ibm939A) IBM-930A (ibm930A, Cp930A) IBM-33722A (Cp33722A, ibm33722A) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-SJIS_0213 () IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-859 (Cp859, ibm859) IBM-837 (ibm837, Cp837) </p>

Platform	Supported Encodings (not in Common Encodings table)
Windows	<p> IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-33722C (ibm-eucjp, Cp33722c) IBM-301 (Cp301, ibm301) IBM-300 (Cp300, ibm300) IBM-290 (ibm290, Cp290) IBM-1399 (ibm1399, Cp1399) IBM-1390 (Cp1390, ibm1390) IBM-1388 (Cp1388, ibm1388) IBM-1385 (Cp1385, ibm1385) IBM-1382 (ibm1382, Cp1382) IBM-1088 (Cp1088, ibm1088) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1115 (Cp1115, ibm1115) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) IBM-1364 (Cp1364, ibm1364) IBM-1370 (Cp1370, ibm1370) IBM-1371 (Cp1371, ibm1371) IBM-1380 (Cp1380, ibm1380) IBM-867 (Cp867, ibm867) IBM-897 (Cp897, ibm897) IBM-924 (Cp924, ibm924) IBM-927 (ibm927, Cp927) IBM-932 (ibm932, Cp932) IBM-947 (Cp947, ibm947) IBM-951 (Cp951, ibm951) IBM-954 (ibm954, Cp954) IBM-971 (Cp971, ibm971) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) </p>

Platform	Supported Encodings (not in Common Encodings table)
Red Hat Enterprise Linux on System x	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) KOI8-RU (ibm-1168, koi8_ru) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM1047 (cp1047, 1047, ibm-1047) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) x-Big5-Solaris (Big5_Solaris) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-IBM964 (964, cp964, ibm-964, ibm964) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-iso-8859-11 (iso-8859-11, iso8859_11) x-JISAutoDetect (JISAutoDetect) x-MS932_0213 () x-MS950-HKSCS (MS950_HKSCS) x-PCK (pck) x-IBM1363C (ibm1363c, cp1363c, ibm-1363c) x-IBM420S (420s, ibm-420s, csibm420s, ibm420s, cp420s) x-IBM864S (csibm864s, ibm864s, cp864s, 864s, ibm-864s) x-IBM943C (cp943c, 943c, ibm-943c, ibm943c) x-IBM949C (ibm949c, cp949c, 949c, ibm-949c) x-IBM954C (cp954c, 954c, ibm-954c, ibm954c) x-ISO-8859-6S (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) x-JIS0208 (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) x-KSC5601 (ksc5601) x-MacDingbat (macdingbat) x-MacHebrew (machebrew) x-MacThai (macthai) x-MacUkraine (macukraine) x-IBM1046S (ibm-1046s, 1046s, cp1046s, ibm1046s) x-IBM-udcJP (IBM-udcJP) JIS_X0201 (jis_x0201, x0201, cshalfwidthkatakana, jis0201) IBM-939A (Cp939A, ibm939A) IBM-930A (ibm930A, Cp930A) IBM-33722A (Cp33722A, ibm33722A) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-SJIS_0213 () IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-859 (Cp859, ibm859) </p>

Platform	Supported Encodings (not in Common Encodings table)
Red Hat Enterprise Linux on System x	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-33722C (ibm-eucjp, Cp33722c) IBM-301 (Cp301, ibm301) IBM-300 (Cp300, ibm300) IBM-290 (ibm290, Cp290) IBM-1399 (ibm1399, Cp1399) IBM-1390 (Cp1390, ibm1390) IBM-1388 (Cp1388, ibm1388) IBM-1385 (Cp1385, ibm1385) IBM-1382 (ibm1382, Cp1382) IBM-1088 (Cp1088, ibm1088) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) IBM-1364 (Cp1364, ibm1364) IBM-1370 (Cp1370, ibm1370) IBM-1371 (Cp1371, ibm1371) IBM-1380 (Cp1380, ibm1380) IBM-867 (Cp867, ibm867) IBM-897 (Cp897, ibm897) IBM-924 (Cp924, ibm924) IBM-927 (ibm927, Cp927) IBM-932 (ibm932, Cp932) IBM-947 (Cp947, ibm947) IBM-951 (Cp951, ibm951) IBM-954 (ibm954, Cp954) IBM-971 (Cp971, ibm971) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) </p>

Platform	Supported Encodings (not in Common Encodings table)
<p>z/OS z/OS z/OS</p>	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) UTF-16 (UTF16, Unicode, UTF_16, UCS-2) Shift_JIS () PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) MacUkraine () MacThai () MacHebrew () MacDingbat () KSC5601 () KOI8-U (koi8_u, ibm-1167) KOI8-RU (ibm-1168, koi8_ru) Johab (x-johab) JIS0208 () JIS0201 () ISO-8859-6S (iso8859-6S, iso8859_6S) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) IBM-971 (Cp971, ibm971) IBM-964 (ibm-euctw, Cp964) IBM-954C (Cp954c) IBM-954 (ibm954, Cp954) IBM-951 (Cp951, ibm951) IBM-949C (Cp949C, ibm949C) IBM-947 (Cp947, ibm947) IBM-943C (ibm943C, Cp943C) IBM-942C (Cp942C, ibm942C) IBM-939 (Cp5035, 5035) IBM-932 (ibm932, Cp932) IBM-930 (Cp5026, 5026) IBM-927 (ibm927, Cp927) IBM-924 (Cp924, ibm924) IBM-918 (ibm918, Cp918) IBM-897 (Cp897, ibm897) IBM-867 (Cp867, ibm867) IBM-1380 (Cp1380, ibm1380) IBM-1371 (Cp1371, ibm1371) IBM-1370 (Cp1370, ibm1370) IBM-1364 (Cp1364, ibm1364) IBM-1363C (ibm1363C, Cp1363C) IBM-1363 (ibm1363, Cp1363) IBM-1088 (Cp1088, ibm1088) IBM-1382 (ibm1382, Cp1382) IBM-1385 (Cp1385, ibm1385) IBM-1386 (ibm1386, Cp1386) IBM-1388 (Cp1388, ibm1388) IBM-1390 (Cp1390, ibm1390) IBM-1399 (ibm1399, Cp1399) IBM-290 (ibm290, Cp290) IBM-300 (Cp300, ibm300) IBM-301 (Cp301, ibm301) IBM-33722 (5050, Cp5050) IBM-33722C (ibm-eucjp, Cp33722c) IBM-930A (ibm930A, Cp930A) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-864S (ibm864S, Cp864S) IBM-859 (Cp859, ibm859) IBM-858 (Cp858, ibm858) </p>

Platform	Supported Encodings (not in Common Encodings table)
  z/OS	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-420S (Cp420S, ibm420S) IBM-1047 (Cp1047, ibm1047) IBM-1046S (ibm1046S, Cp1046S) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1140 (ibm1140, Cp1140) IBM-1141 (Cp1141, ibm1141) IBM-1142 (Cp1142, ibm1142) IBM-1143 (Cp1143, ibm1143) IBM-1144 (ibm1144, Cp1144) IBM-1145 (Cp1145, ibm1145) IBM-1146 (Cp1146, ibm1146) IBM-1147 (Cp1147, ibm1147) IBM-1148 (ibm1148, Cp1148) IBM-1149 (Cp1149, ibm1149) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) </p>

Platform	Supported Encodings (not in Common Encodings table)
Linux on POWER Systems - Big Endian	<p> windows-1256S (Cp1256s, ibm-1256s) UTF-8J (UTF8J) UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) UTF-16 (UTF16, Unicode, UTF_16, UCS-2) Shift_JIS () PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) MacUkraine () MacThai () MacHebrew () MacDingbat () KSC5601 () KOI8-U (koi8_u, ibm-1167) KOI8-RU (ibm-1168, koi8_ru) Johab (x-johab) JIS0208 () JIS0201 () ISO-8859-6S (iso8859-6S, iso8859_6S) ISO-8859-16 (8859-16, iso8859_16, iso8859-16) ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) IBM-971 (Cp971, ibm971) IBM-964 (ibm-euctw, Cp964) IBM-954C (Cp954c) IBM-954 (ibm954, Cp954) IBM-951 (Cp951, ibm951) IBM-949C (Cp949C, ibm949C) IBM-947 (Cp947, ibm947) IBM-943C (ibm943C, Cp943C) IBM-942C (Cp942C, ibm942C) IBM-939 (Cp5035, 5035) IBM-932 (ibm932, Cp932) IBM-930 (Cp5026, 5026) IBM-927 (ibm927, Cp927) IBM-924 (Cp924, ibm924) IBM-918 (ibm918, Cp918) IBM-897 (Cp897, ibm897) IBM-867 (Cp867, ibm867) IBM-1380 (Cp1380, ibm1380) IBM-1371 (Cp1371, ibm1371) IBM-1370 (Cp1370, ibm1370) IBM-1364 (Cp1364, ibm1364) IBM-1363C (ibm1363C, Cp1363C) IBM-1047 (Cp1047, ibm1047) IBM-1088 (Cp1088, ibm1088) IBM-1382 (ibm1382, Cp1382) IBM-1385 (Cp1385, ibm1385) IBM-1386 (ibm1386, Cp1386) IBM-1388 (Cp1388, ibm1388) IBM-1390 (Cp1390, ibm1390) IBM-1399 (ibm1399, Cp1399) IBM-290 (ibm290, Cp290) IBM-300 (Cp300, ibm300) IBM-301 (Cp301, ibm301) IBM-33722 (5050, Cp5050) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) IBM-864S (ibm864S, Cp864S) IBM-859 (Cp859, ibm859) IBM-858 (Cp858, ibm858) </p>

Platform	Supported Encodings (not in Common Encodings table)
Linux on POWER Systems - Big Endian	<p> IBM-837 (ibm837, Cp837) IBM-836 (ibm836, Cp836) IBM-835 (ibm835, Cp835) IBM-833 (ibm833, Cp833) IBM-808 (Cp808, ibm808) IBM-720 (Cp720, ibm720) IBM-420S (Cp420S, ibm420S) IBM-33722C (ibm-eucjp, Cp33722c) IBM-1046S (ibm1046S, Cp1046S) IBM-1043 (Cp1043, ibm1043) IBM-1041 (Cp1041, ibm1041) IBM-1027 (Cp1027, ibm1027) CESU-8 (CESU8) COMPOUND_TEXT (x-compound-text, x11-compound-text) GB2312 (gb2312-1980, gb2312-80) GBK (GBK) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1114 (Cp1114, ibm1114) IBM-1115 (Cp1115, ibm1115) IBM-1140 (ibm1140, Cp1140) IBM-1141 (Cp1141, ibm1141) IBM-1142 (Cp1142, ibm1142) IBM-1143 (Cp1143, ibm1143) IBM-1144 (ibm1144, Cp1144) IBM-1145 (Cp1145, ibm1145) IBM-1146 (Cp1146, ibm1146) IBM-1147 (Cp1147, ibm1147) IBM-1148 (ibm1148, Cp1148) IBM-1149 (Cp1149, ibm1149) IBM-1351 (Cp1351, ibm1351) IBM-1362 (Cp1362, ibm1362) IBM-1363 (ibm1363, Cp1363) </p>

Platform	Supported Encodings (not in Common Encodings table)
HP (PA-RISC)	<p> UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE) UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE) IBM01147 (ccsid01147, cp1147, 1147, cp01147) IBM01148 (cp1148, ccsid01148, 1148, cp01148) IBM01149 (cp1149, cp01149, ccsid01149, 1149) IBM1047 (cp1047, 1047, ibm-1047) IBM918 (cp918, ebcdic-cp-ar2, ibm-918, 918) ISO-2022-JP-2 (csISO2022JP2, iso2022jp2) Roman9 (Roman9) x-Big5-Solaris (Big5_Solaris) x-eucJP-Open (EUC_JP_Solaris, eucJP-open) x-IBM33722 (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) x-IBM930 (cp930, ibm930, ibm-930, 930) x-IBM939 (ibm-939, ibm939, cp939, 939) x-windows-iso2022jp (windows-iso2022jp) x-windows-50221 (ms50221, cp50221) x-windows-50220 (cp50220, ms50220) X-UTF-32LE-BOM (UTF_32LE_BOM, UTF-32LE-BOM) X-UTF-32BE-BOM (UTF_32BE_BOM, UTF-32BE-BOM) x-SJIS_0213 () x-PCK (pck) x-MS950-HKSCS (MS950_HKSCS) x-MS932_0213 () x-JISAutoDetect (JISAutoDetect) x-iso-8859-11 (iso-8859-11, iso8859_11) x-ISO-2022-CN-CNS (ISO-2022-CN-CNS, ISO2022CN_CNS) x-IBM964 (964, cp964, ibm-964, ibm964) IBM01146 (ccsid01146, cp01146, cp1146, 1146) IBM01145 (cp1145, cp01145, ccsid01145, 1145) IBM01144 (cp01144, cp1144, ccsid01144, 1144) IBM01143 (cp01143, 1143, ccsid01143, cp1143) IBM01142 (cp01142, cp1142, 1142, ccsid01142) IBM01141 (cp1141, ccsid01141, cp01141, 1141) IBM01140 (ccsid01140, cp01140, 1140, cp1140) IBM00858 (cp858, ccsid00858, 858, cp00858) X-UnicodeLittle (UnicodeLittle) X-UnicodeBig (UnicodeBig) COMPOUND_TEXT (x-compound-text, x11-compound-text) hp-roman8 (roman8, ibm-1051, r8, Cp1051) IBM-1364 (Cp1364, ibm1364) IBM-942C (Cp942C, ibm942C) IBM-943C (ibm943C, Cp943C) IBM-949C (Cp949C, ibm949C) JIS0201 () JIS0208 () KOI8-U (koi8_u, ibm-1167) MacDingbat () MacHebrew () MacThai () MacUkraine () UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4) </p>

Platforms by Encoding

Encoding	Aliases	Platforms on which this encoding is supported
x-MacUkraine	macukraine	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x

Encoding	Aliases	Platforms on which this encoding is supported
x-MacThai	macthai	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-MacHebrew	machebrew	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-MacDingbat	macdingbat	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-KSC5601	ksc5601	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-JIS0208	jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-ISO-8859-6S	8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM954C	cp954c, 954c, ibm-954c, ibm954c	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM949C	ibm949c, cp949c, 949c, ibm-949c	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x

Encoding	Aliases	Platforms on which this encoding is supported
x-IBM943C	cp943c, 943c, ibm-943c, ibm943c	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM864S	csibm864s, ibm864s, cp864s, 864s, ibm-864s	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM420S	420s, ibm-420s, csibm420s, ibm420s, cp420s	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM1363C	ibm1363c, cp1363c, ibm-1363c	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM1046S	ibm-1046s, 1046s, cp1046s, ibm1046s	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
x-IBM-udcJP	IBM-udcJP	SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
JIS_X0201	jis_x0201, x0201, cshalfwidthkatakana, jis0201	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
IBM-939A	Cp939A, ibm939A	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
IBM-930A	ibm930A, Cp930A	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x,  z/OS

Encoding	Aliases	Platforms on which this encoding is supported
  IBM-924_LF	Cp924_LF, ibm924_LF	IBM i
IBM-33722A	Cp33722A, ibm33722A	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x
  IBM-1141_LF	Cp1141_LF, ibm1141_LF	IBM i
  IBM-1047_LF	Cp1047_LF, ibm1047_LF	IBM i
x-windows-iso2022jp	windows-iso2022jp	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-windows-50221	ms50221, cp50221	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-windows-50220	cp50220, ms50220	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
X-UTF-32LE-BOM	UTF_32LE_BOM, UTF-32LE-BOM	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
X-UTF-32BE-BOM	UTF_32BE_BOM, UTF-32BE-BOM	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-SJIS_0213		Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-PCK	pck	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-MS950-HKSCS	MS950_HKSCS	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-MS932_0213		Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-JISAutoDetect	JISAutoDetect	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-iso-8859-11	iso-8859-11, iso8859_11	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-ISO-2022-CN-CNS	ISO-2022-CN-CNS, ISO2022CN_CNS	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM964	964, cp964, ibm-964, ibm964	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM939	ibm-939, ibm939, cp939, 939	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM930	cp930, ibm930, ibm-930, 930	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM33722	ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-eucJP-Open	EUC_JP_Solaris, eucJP-open	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-Big5-Solaris	Big5_Solaris	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
Roman9	Roman9	HP (PA-RISC)
ISO-2022-JP-2	csISO2022JP2, iso2022jp2	Solaris, SUSE Linux Enterprise Server on System x, Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM918	cp918, ebcdic-cp-ar2, ibm-918, 918	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM1047	cp1047, 1047, ibm-1047	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01149	cp1149, cp01149, ccsid01149, 1149	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM01148	cp1148, ccsid01148, 1148, cp01148	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01147	ccsid01147, cp1147, 1147, cp01147	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01146	ccsid01146, cp01146, cp1146, 1146	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01145	cp1145, cp01145, ccsid01145, 1145	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01144	cp01144, cp1144, ccsid01144, 1144	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01143	cp01143, 1143, ccsid01143, cp1143	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM01142	cp01142, cp1142, 1142, ccsid01142	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01141	cp1141, ccsid01141, cp01141, 1141	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01140	ccsid01140, cp01140, 1140, cp1140	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM00858	cp858, ccsid00858, 858, cp00858	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for IBM Z Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
X-UnicodeLittle	UnicodeLittle	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z  AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
X-UnicodeBig	UnicodeBig	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
IBM-864S	ibm864S, Cp864S	 IBM i,  AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-859	Cp859, ibm859	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-858	Cp858, ibm858	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-837	ibm837, Cp837	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-836	ibm836, Cp836	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-835	ibm835, Cp835	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-833	ibm833, Cp833	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-808	Cp808, ibm808	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-720	Cp720, ibm720	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-420S	Cp420S, ibm420S	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-33722C	ibm-eucjp, Cp33722c	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-33722	5050, Cp5050	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-301	Cp301, ibm301	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-300	Cp300, ibm300	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-290	ibm290, Cp290	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1399	ibm1399, Cp1399	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1390	Cp1390, ibm1390	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1388	Cp1388, ibm1388	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1386	ibm1386, Cp1386	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1385	Cp1385, ibm1385	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1382	ibm1382, Cp1382	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1088	Cp1088, ibm1088	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1047	Cp1047, ibm1047	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1046S	ibm1046S, Cp1046S	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1043	Cp1043, ibm1043	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1041	Cp1041, ibm1041	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1027	Cp1027, ibm1027	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
CESU-8	CESU8	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
COMPOUND_TEXT	x-compound-text, x11-compound-text	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
GB2312	gb2312-1980, gb2312-80	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
GBK	GBK	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
hp-roman8	roman8, ibm-1051, r8, Cp1051	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
IBM-1114	Cp1114, ibm1114	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1115	Cp1115, ibm1115	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1140	ibm1140, Cp1140	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1141	Cp1141, ibm1141	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1142	Cp1142, ibm1142	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1143	Cp1143, ibm1143	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1144	ibm1144, Cp1144	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1145	Cp1145, ibm1145	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1146	Cp1146, ibm1146	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1147	Cp1147, ibm1147	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1148	ibm1148, Cp1148	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1149	Cp1149, ibm1149	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1351	Cp1351, ibm1351	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1362	Cp1362, ibm1362	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1363	ibm1363, Cp1363	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1363C	ibm1363C, Cp1363C	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-1364	Cp1364, ibm1364	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1370	Cp1370, ibm1370	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1371	Cp1371, ibm1371	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-1380	Cp1380, ibm1380	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-867	Cp867, ibm867	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-897	Cp897, ibm897	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-918	ibm918, Cp918	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-924	Cp924, ibm924	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-927	ibm927, Cp927	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-930	Cp5026, 5026	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-932	ibm932, Cp932	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-939	Cp5035, 5035	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-942C	Cp942C, ibm942C	Solaris,  IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
IBM-943C	ibm943C, Cp943C	 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
IBM-947	Cp947, ibm947	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
IBM-949C	Cp949C, ibm949C	 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
IBM-951	Cp951, ibm951	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-954	ibm954, Cp954	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
IBM-954C	Cp954c	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-964	ibm-euctw, Cp964	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
IBM-971	Cp971, ibm971	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
ISO-8859-10	latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
ISO-8859-14	ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
ISO-8859-16	8859-16, iso8859_16, iso8859-16	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
ISO-8859-6S	iso8859-6S, iso8859_6S	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
JIS0201		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
JIS0208		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
Johab	x-johab	 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
KOI8-RU	ibm-1168, koi8_ru	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
KOI8-U	koi8_u, ibm-1167	Solaris,  IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
KSC5601		 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
MacDingbat		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
MacHebrew		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
MacThai		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
MacUkraine		 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
PTCP154	PT154, IBM-1169, Cyrillic-Asian, csPTCP154	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian
Shift_JIS		 IBM i, AIX,  z/OS, Linux on POWER Systems - Big Endian
UTF-16	UTF16, Unicode, UTF_16, UCS-2	 IBM i, HP-UX (Itanium), AIX,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
UTF-32	UCS-4, UTF32, ISO-10646-UCS-4	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
UTF-32BE	UTF_32BE, X-UTF-32BE, UTF32BE	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
UTF-32LE	UTF_32LE, X-UTF-32LE, UTF32LE	Solaris, SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian , HP (PA-RISC)
UTF-8J	UTF8J	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Encoding	Aliases	Platforms on which this encoding is supported
windows-1256S	Cp1256s, ibm-1256s	SUSE Linux Enterprise Server on System x,  IBM i, Linux for IBM Z AIX, Windows, Red Hat Enterprise Linux on System x,  z/OS, Linux on POWER Systems - Big Endian

Related concepts

[“Using transfer definition files” on page 152](#)

You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Related reference

[“Transferring text files with MFT” on page 617](#)

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

How MFT agents use Java heap and native heap memory

An IBM MQ Managed File Transfer agent runs as a Java process. As such, the agent runs in the virtualized environment of the Java Virtual Machine (JVM).

The JVM itself is a native process, is bounded by the hardware and operating system. The JVM maintains two memory areas:

- The Java heap

This contains the instances of Java objects and is managed by garbage collection processing. The maximum size of the Java heap is allocated during JVM startup using the **-Xmx** JVM option.

- The native heap

The native heap contains resources for the JVM itself; for example, the Just-In-Time Compiler, Classes, and ClassLoaders.

An agent primarily uses the Java heap. When performing managed transfers, the agent uses the Java heap to create Java objects that are required for the transfer. Any file data that is read into buffers by the agent is also stored in Java heap memory.

An agent does not itself contain any code that uses the native heap. However, there is native code in the Java message queuing interface (JMQUI) that the agent uses to communicate with its agent queue manager.

This native code is used when an agent connects to its agent queue manager using the BINDINGS transport. This is a local shared memory connection (sometimes referred to as interprocess communication, or IPC), rather than a TCP/IP connection which is used if an agent connects using the CLIENT transport. When an agent is configured to use the BINDINGS transport, the native heap is used to pass messages and commands between the agent and the agent queue manager.

This means that a heavily loaded agent that is connected to its agent queue manager using the BINDINGS transport makes more extensive use of the native heap, when compared to an equivalent agent which is connected using the CLIENT transport.

One common misconception is that the Java heap for an agent must be equal to (or greater than) the size of the largest file that is to be transferred. This is not correct as file data is read into memory in stages.

As a guide, the maximum amount of Java heap that is used to store file data for each transfer can be roughly calculated as follows:

```
Memory allocated for a transfer = agentCheckpointInterval *  
agentFrameSize * agentWindowSize * agentChunkSize
```

How Java heap and native heap usage affects agents

When a `java.lang.OutOfMemoryError` occurs, you might think it reasonable to increase the amount of Java heap available to the application, using the `-Xmx` Java System Property. For example, the following property setting attempts to allocate a maximum Java heap size of 2GB:

```
-Xmx2048M
```

However, allocating too much Java heap for an application can cause a `java.lang.OutOfMemoryError` to occur, due to native heap exhaustion. This is because, as the Java heap space grows, the native heap must shrink to accommodate it.

For information about how to prevent `java.lang.OutOfMemoryErrors` that are caused by native heap exhaustion, see [What to do if your MFT agent ABENDS with a java.lang.OutOfMemoryError due to native memory exhaustion](#).

XML message formats used by MFT

Managed File Transfer uses messages in XML format for a number of purposes: to command an agent; to log information about the monitors, schedules, and transfers; and to define information used for configuration. The logical structure of the XML formats used for these purposes described by XML schema.

Each version of Managed File Transfer uses an XML schema to validate messages written in XML. The agent extracts the XML schema version and determines whether the schema is supported.

After you have installed Managed File Transfer, you can find the Managed File Transfer message schema files in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/schema`. The following schemas are included:

Schemas for XML messages that can be put on an agent command queue

- FileTransfer.xsd
- Internal.xsd
- Monitor.xsd
- PingAgent.xsd

For more information about putting XML messages on an agent command queue, see [“Controlling MFT by putting messages on the agent command queue”](#) on page 258.

Schemas for XML messages that are published to the SYSTEM.FTE topic

- MonitorList.xsd
- MonitorLog.xsd
- ScheduleList.xsd
- ScheduleLog.xsd
- TransferLog.xsd
- TransferStatus.xsd

For more information about XML messages that are published to the SYSTEM.FTE topic and the structure of the SYSTEM.FTE topic, see [“SYSTEM.FTE topic” on page 594](#).

Other schemas used by Managed File Transfer

`fteutils.xsd`. This schema contains common element definitions and is included by some of the other schemas.

`Notification.xsd`

`ProtocolBridgeCredentials.xsd`

`ProtocolBridgeProperties.xsd`

`ConnectDirectCredentials.xsd`

`ConnectDirectNodeProperties.xsd`

`ConnectDirectProcessDefinitions.xsd`

`Reply.xsd`

`UserSandboxes.xsd`

MFT agent status message format

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

The following information is included:

- Agent name
- Platform the agent is running on
- Agent description (if provided)
- Agent's queue manager
- Time zone that the agent is running in
- Agent version
- Agent transfer limits
- State of each of the agent's current transfers. These states are listed in [Agent transfer states](#)
- Type of agent

If the agent is a protocol bridge agent the following information is also included:

- Type of protocol bridge agent
- Host name or IP address of the protocol bridge server

The agent status is republished whenever the agent transfer states change, but by default no more than every 30 seconds. You can change this default setting using the `agentStatusPublishRateLimit` agent property, which is described in: [Advanced agent properties](#).

The following example output shows the keys used for each data element in the agent status:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="agentOsName">Windows 7</entry>
  <entry key="agentDescription"/>
  <entry key="queueManager">QM1</entry>
  <entry key="agentTimeZone">Europe/London</entry>
  <entry key="agentVersion">1.00</entry>
  <entry key="agentName">FTEAGENT</entry>
  <entry key="maxDestinationTransfers">25</entry>
  <entry key="maxSourceTransfers">25</entry>
  <entry key="maxQueuedTransfers">100</entry>
  <entry
key="DestinationTransferStates">414d51204d554e474f202020202020d857374a60a72622=RunningTransfer
414d51204d554e474f202020202020d857374a69a72622=RunningTransfer
414d51204d554e474f202020202020d857374a75a72622=RunningTransfer
  </entry>
```

```

    <entry
key="SourceTransferStates">414d51204d554e474f202020202020d857374a93a72622=NegotiatingTransfer
414d51204d554e474f202020202020d857374a78a72622=RunningTransfer
414d51204d554e474f202020202020d857374aaba72622=NewSenderTransfer
414d51204d554e474f202020202020d857374a63a72622=RunningTransfer
    </entry>
</properties>

```

The following example output shows the keys used for each data element in the agent status of a protocol bridge agent:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="agentOsName">Windows 7</entry>
  <entry key="agentDescription"/>
  <entry key="queueManager">QM1</entry>
  <entry key="agentTimeZone">Europe/London</entry>
  <entry key="agentVersion">1.00</entry>
  <entry key="agentName">BRIDGE</entry>
  <entry key="protocolBridgeType">ftp</entry>
  <entry key="protocolBridgeServerHost">ftpserver.example.org</entry>
  <entry key="maxDestinationTransfers">25</entry>
  <entry key="maxSourceTransfers">25</entry>
  <entry key="maxQueuedTransfers">100</entry>
  <entry key="DestinationTransferStates">414d51204d554e474f2020202020d857374a60a72622=RunningTransfer
  </entry>
  <entry key="SourceTransferStates">414d51204d554e474f2020202020d857374a93a72622=NegotiatingTransfer
  </entry>
</properties>

```

MFT agent transfer states

A Managed File Transfer Agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent.

Transfer state	Explanation
NewSenderTransfer	A new transfer from the source agent that the negotiation has not started for.
NewReceiverTransfer	A new transfer has been created at the destination agent as part of negotiation, but the transfer is not yet running.
NegotiatingTransfer	A source agent is in negotiation with the destination agent before running a transfer.
RunningTransfer	A transfer from either a source agent or destination agent that is in the normal running state
RecoveringTransfer	When either a source or destination agent starts the recovery process, any transfers in running state are moved into transfer state. Transfers are moved out of this state into ReSynchronisingTransfer state when a resynchronization message is sent to the peer agent. For example, if the destination agent starts the recovery process for a running transfer, the transfer is moved into the ReSynchronisingTransfer state when a resynchronization message is sent to its source agent.
ReSynchronisingTransfer	A transfer source or destination agent has found a problem and has sent a resynchronization message to its respective destination or source agent.
CompletedTransfer	A destination agent has completed the transfer and has sent a completion message to the source agent. The destination agent is waiting for an acknowledgment message from the source agent.

Transfer state	Explanation
CompleteReceivedTransfer	A source agent has received a completion message from the destination agent and has sent a message back to the destination agent to acknowledge the completion message.
CancelledNewTransfer	A source agent has received a cancel message for a new transfer.
CancelledInProgressTransfer	A source agent has received a cancel message for an in-progress transfer.
ResumingTransfer	A source agent has received a resynchronize response message and now schedules the transfer to restart.
RestartingTransfer	A source or destination agent has received a resynchronize request message and is waiting for the respective destination or source agent to restart.
WaitingForDestinationCapacity	A source agent has received a DESTINATION_CAPACITY_EXCEEDED error from the destination agent. The transfer is now in a waiting state to be retried after a period.
FailedTransferEnding	The transfer has failed but the completion log message has not been published and the transfer has not been removed from the state store. For example, this state can occur if an agent process is stopped after a failure response has been received from the destination agent but before the subsequent processing has been completed.

MFT monitor list message format

The XML messages that are published as retained publications to the topic string `SYSTEM.FTE/monitors/agent_name/monitor_name` conform to the `MonitorList.xsd` schema. Each XML message lists an active monitor belonging to that agent. This information is used by the `ftelListMonitors` command and the IBM MQ Explorer plug-in to display a list of monitors to the user. The `MonitorList.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `MonitorList.xsd` schema imports `Monitor.xsd`, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

  <xsd:include schemaLocation="Monitor.xsd"/>

  <xsd:element name="monitorList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="status" type="monitorStatusType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="configuration" type="monitorConfigurationType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="pollInterval" type="pollIntervalType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="batch" type="batchType" minOccurs="1" maxOccurs="1"/>
        <xsd:any minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="agent" type="xsd:string" use="required"/>
      <xsd:attribute name="monitor" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="monitorStatusType">
    <xsd:sequence>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="state" type="xsd:token"/>
    <xsd:anyAttribute/>
  </xsd:complexType>

  <xsd:complexType name="monitorConfigurationType">
    <xsd:sequence>
      <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="resources" type="monitorResourcesType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="triggerMatch" type="triggerMatchType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```

        <xsd:element name="tasks" type="monitorListTasksType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
</xsd:complexType>

<xsd:complexType name="monitorListTasksType">
    <xsd:sequence>
        <xsd:element name="task" type="monitorListTaskType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorListTaskType">
    <xsd:sequence>
        <xsd:element name="name" type="monitorTaskNameType" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="taskXML" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

</xsd:schema>

```

Understanding the monitor list message

The elements and attributes used in the monitor list messages are described in the following list:

<monitorList>

Group element containing the elements describe a monitor that is defined for the agent.

Attribute	Description
agent	Required. The name of the agent that the resource monitor is defined on.
monitor	Required. The name of the monitor. Unique for this agent.
version	Required. The version of the monitor list message format.

<status>

The status of the monitor.

Attribute	Description
state	The state of the monitor.

<configuration>

Group element containing the elements describe the configuration of the monitor.

<description>

A description of the monitor. (Not currently used.)

<resources>

The resource or resources being monitored.

<directory>

A directory to monitor.

Attribute	Description
recursionLevel	The number of directory levels down from the top level to monitor.
id	The ID of the resource.

<queue>

A queue to monitor.

Attribute	Description
id	The ID of the resource.

<triggerMatch>

Element that contains the <conditions> element.

<conditions>

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain only one of the following elements: <allOf>, <anyOf>, or <condition>.

<allOf>

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered all of the conditions inside of this element must be met.

<anyOf>

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered only one of the conditions inside of this element must be met.

<condition>

Element that contains a single condition that the resource monitor is monitoring for. This element can contain only one of the following elements: <fileMatch>, <fileNoMatch>, <fileSize>, <queueNotEmpty>, <completeGroups>, or <fileSizeSame>. It can also contain a <name> element and a <resource> element.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:

- queueNotEmpty
- completeGroups

<fileMatch>

Group element for a file name match condition.

<pattern>

Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

<fileNoMatch>

Group element for an inverse file name match condition.

<pattern>

Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

<fileSize>

Group element for a file size comparison.

<compare>

Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: <ul style="list-style-type: none"> • B - bytes • KB - kilobytes • MB - megabytes

Attribute	Description
	<ul style="list-style-type: none"> GB - gigabytes <p>The units value is case insensitive, so mb' works as well as MB'.</p>

<pattern>

File name pattern to match. Default is * (any file will match).

<queueNotEmpty>

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

<completeGroups>

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

<name>

Name of the condition.

<resource>

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

<tasks>

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

<task>

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

<name>

Name of the task. Accepts any alphanumeric characters.

<description>

Description of the task. Any text value is allowed.

<taskXML>

The XML message that describes the task that the monitor is to perform. The contents of this element are in an escaped XML format.

<pollInterval>

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	<p>Specifies the time units for the poll interval. Valid values are:</p> <ul style="list-style-type: none"> seconds minutes hours days weeks months years

<batch>

The maximum number of trigger matches to include in a single batch.

Attribute	Description
maxSize	The maximum number of trigger matches to include in a single batch

The following XML shows an example of a retained publication which is published to the topic string SYSTEM.FTE/monitors/*agent_name*/MONITORTWO when the monitor called MONITORTWO is created on AGENT_JUPITER. The escaped XML within the <taskXML> element describes the task that is submitted when the monitor condition is met.

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:lst="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd"
  version="4.00"
  agent="AGENT_JUPITER"
  monitor="MONITORTWO">
  <status state="started"/>
  <configuration>
    <description/>
    <resources>
      <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
      <conditions>
        <condition>
          <name/>
          <resource id=""/>
          <fileMatch>
            <pattern>*.completed</pattern>
          </fileMatch>
        </condition>
      </conditions>
    </triggerMatch>
    <tasks>
      <task>
        <name/>
        <description/>
        <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
          xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
            &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
            &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
            &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
            &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
            &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
              &lt;source disposition="leave" recursive="false"&gt;&lt;file
                &gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
              &lt;destination exist="error" type="directory"&gt;
                &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
              &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;
            &lt;/request&gt;
          </taskXML>
        </task>
      </tasks>
    </configuration>
    <pollInterval units="minutes">1</pollInterval>
    <batch maxSize="1"/>
  </lst:monitorList>
```

MFT schedule list message format

The XML message that is published to a retained publication to the topic string SYSTEM.FTE/Scheduler/*agent_name* conforms to the ScheduleList.xsd schema. This XML message lists all active schedules belonging to that agent. This information is used by the **fteListScheduledTransfers** command and the IBM MQ Explorer to display a list of schedules to the user. The ScheduleList.xsd schema document is located in the *MQ_INSTALLATION_PATH/mqft/samples/schema* directory. The ScheduleList.xsd schema imports FileTransfer.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="FileTransfer.xsd"/>
  <xsd:element name="schedules">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="managedTransfer" type="scheduledManagedTransferType" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="required"/>
      <xsd:attribute name="agent" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="scheduledManagedTransferType">
    <xsd:sequence>
```

```

<xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
<xsd:element name="schedule" type="scheduleListType" maxOccurs="1" minOccurs="0"/>
<xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
<xsd:element name="destinationAgent" type="agentClientType" maxOccurs="1" minOccurs="1"/>
<xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0"/>
<xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
<xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
<xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="id" type="idType" use="required"/>
</xsd:complexType>

<xsd:complexType name="scheduleListType">
  <xsd:sequence>
    <xsd:element name="submit" type="submitType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="repeat" type="repeatType" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="next" type="noZoneTimeType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

Understanding the schedule list message

The elements and attributes used in the schedule list messages are described in the following list:

<schedules>

Group element containing information about all of the schedules defined on a single agent.

Attribute	Description
agent	Required. The name of the source agent that the schedule is defined on.
size	Required. The number of schedules defined on this agent.
version	Required. The version of the schedule list message format.

<managedTransfer>

Group element containing information about a single schedule.

Attribute	Description
id	Required. The hexadecimal string ID of the schedule request message.

<originator>

The originator of the schedule request.

<hostName>

The host name of the machine that the schedule request was submitted from.

<userID>

The user ID of the user that submitted the schedule request.

<mqmdUserID>

The MQMD user ID of the user that submitted the schedule request.

<schedule>

Element that contains the elements that describe when the scheduled transfer occurs.

<submit>

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. The value of this attribute can be one of the following values: <ul style="list-style-type: none"> source - use the time zone of the source agent admin - use the time zone of the administrator issuing the command UTC - use Coordinated Universal Time
timezone	The time zone description according to the timebase value

<repeat>

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> • minutes • hours • days • weeks • months • years

<frequency>

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> • minutes • hours • days • weeks • months • years

<expireTime>

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

<expireCount>

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

<next>

Specifies the date and time when the next scheduled transfer is due to start.

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<trigger>

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The following are valid values: <ul style="list-style-type: none">• yes - log entries are created for failed triggered transfers• no - log entries are not created for failed triggered transfers

<reply>

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file or the default of `WMQFTE.*` if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

<transferSet>

Specifies a group of file transfers you want the scheduled transfer to perform together. During transmission `<transferSet>` is a group element containing `<item>` elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

<job>

Optional group element containing job information for the entire transfer specification. `<job>` is a user-defined job name identifier that is added to the log message when the transfer has started. This `<job>` element is the same as the `<job>` element that appears in the transfer log message, which is described in the following topic: [“File transfer log message formats”](#) on page 723.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<schedules xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  size="2"
  version="4.00"
  agent="AGENT_JUPITER"
  xsi:noNamespaceSchemaLocation="ScheduleList.xsd">
  <managedTransfer id="1">
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00+0000</
submit>
      <next>2010-01-01T21:00+0000</next>
    </schedule>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <destinationAgent agent="AGENT_SATURN" QMgr="QM_JUPITER"/>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E06</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/etc/passwd</file>
        </source>
        <destination type="directory" exist="overwrite">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</schedules>
```

```

        </destination>
      </item>
    </transferSet>
  </managedTransfer>
  <managedTransfer id="2">
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-12-31T09:00+0000</
submit>
      <next>2010-12-31T09:00+0000</next>
    </schedule>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <destinationAgent agent="AGENT_NEPTUNE" QMgr="QM_JUPITER"/>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E09</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/etc/passwd</file>
        </source>
        <destination type="directory" exist="overwrite">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</schedules>

```

MFT example template XML message

When a template is created, a message is published to the SYSTEM.FTE topic with a topic string of Templates/*template_ID*. This example XML describes a single template defined in your Managed File Transfer network.

```

<?xml version="1.0" encoding="UTF-8"?>
<transferTemplate version="4.00" id="baf9df73-45c2-4bb0-a085-292232ab66bc">
  <name>BASIC_TEMPLATE</name>
  <sourceAgentName>AGENT_JUPITER</sourceAgentName>
  <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
  <destinationAgentName>AGENT_SATURN</destinationAgentName>
  <destinationAgentQMgr>QM_JUPITER</destinationAgentQMgr>
  <fileSpecs>
    <item mode="binary" checksumMethod="MD5">
      <source recursive="false" disposition="leave">
        <file>/etc/passwd</file>
      </source>
      <destination type="directory" exist="overwrite">
        <file>/tmp</file>
      </destination>
    </item>
  </fileSpecs>
  <priority>0</priority>
</transferTemplate>

```

Related tasks

[“Creating a file transfer template using IBM MQ Explorer” on page 188](#)

You can create a file transfer template from IBM MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference

[“fteCreateTemplate: create new file transfer template” on page 401](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

File transfer status message format

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message

is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the *MQ_INSTALLATION_PATH/mqft/samples/schema* directory of your WMQMFT installation.

Schema

The following schema describes which elements are valid in a transfer status XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="transaction">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="sourceAgent" type="agentType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destinationAgent" type="agentType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="transferSet" type="transferSetType"
          maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="transferSetType">
    <xsd:sequence>
      <xsd:element name="stats" type="statsType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="current" type="currentType"
        maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="currentType">
    <xsd:sequence>
      <xsd:element name="source" type="fileSourceType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="destination" type="fileDestinationType"
        maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="transferred" type="xsd:nonNegativeInteger"
      use="required"/>
    <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="statsType">
    <xsd:attribute name="bytes" type="xsd:nonNegativeInteger"
      use="required"/>
    <xsd:attribute name="seconds" type="xsd:decimal"
      use="required"/>
    <xsd:attribute name="currentItem" type="xsd:nonNegativeInteger"
      use="required"/>
    <xsd:attribute name="totalItems" type="xsd:nonNegativeInteger" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

Understanding the transfer status message

The elements and attributes used in the transfer status messages are described in the following list:

<transaction>

Group element that contains all of the elements for the file transfers.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

Attribute	Description
ID	The unique identifier for the file transfer.

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

<transferset>

Specifies a group of file transfers being performed together. All of the files in the transfer must originate at the same source agent and end at the same destination agent.

Attribute	Description
time	Specifies the date and time (in date time format).

<stats>

Required. Defines metrics about the transfer, including the number of bytes copied so far, in the given number of seconds. Also supplies the current item number out of the total number of items in the <transferSet>.

Attribute	Description
bytes	Number of bytes copied so far.
seconds	Number of seconds taken to transfer those bytes.
currentItem	The index of the current item being transferred.
totalItems	The total number of items being transferred.

<current>

Optional element. Group element that contains elements that specify the file transfer currently in progress. The <current> element indicates how many bytes of data have been transferred so far for the current item and the expected total number of bytes

<source>

Group element that contains the element specifying the source file name.

<file>

Specifies the source path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form of the path.

<destination>

Group element that contains the element specifying the destination file name or specification.

<file>

Specifies the destination path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form of the path.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
filesystem	Specifies the name of the file space where the destination file is written.

<queue>

When used with the <destination> element, specifies the name of the queue you want to transfer to. This name is in the format QUEUE or QUEUE@QUEUE_MANAGER.

File transfer progress message examples

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Transfers/agent_name/transfer_ID. The XML examples show the progress message for a single file transfer and for a multiple file transfer.

Single file transfer

The following example shows the details of a single file transfer that is in progress.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
  <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <transferSet time="2011-01-26T13:03:26.542Z">
  <stats bytes="1198" seconds="0.018" currentItem="1" totalItems="1"/>
  <current transferred="1151" size="1151">
  <source>
  <file>/etc/passwd</file>
  </source>
  <destination>
  <file>/tmp/passwd</file>
  </destination>
  </current>
  </transferSet>
</transaction>
```

Multiple file transfer

If there were more files in the transfer set, the transfer status message indicates which one is being processed and how many bytes have been transferred so far.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
  <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <transferSet time="2011-01-26T13:12:58.636Z">
  <stats bytes="440" seconds="0.082" currentItem="10" totalItems="10"/>
  <current transferred="0" size="0">
  <source>
  <file>/srv/nfs/incoming/file10.txt</file>
  </source>
  <destination>
  <file>/srv/nfs/outgoing/file10.txt</file>
  </destination>
  </current>
```

```
</transferSet>
</transaction>
```

File transfer log message formats

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_ID`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

If you want to monitor file transfers or collect data about them, set up a subscription to a wildcard topic tailored to the transfers you are interested in. For example:

```
Log/#
```

or,

```
Log/FTEAGENT/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

Schema

The following schema describes which elements are valid in a transfer log XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="transaction">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="action" type="actionType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent" type="agentExitStatusType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceWebGateway" type="webGatewayType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceWebUser" type="webUserType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationAgent" type="agentExitStatusType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationWebGateway" type="webGatewayType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationWebUser" type="webUserType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="agent" type="agentExitStatusType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="status" type="statusType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="scheduleLog" type="scheduleLogType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="statistics" type="statisticsType" maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
      <xsd:attribute name="relatedID" type="IDType" use="optional"/>
      <xsd:attribute name="agentRole" type="agentRoleType" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

```

<xsd:complexType name="agentExitStatusType">
  <xsd:complexContent>
    <xsd:extension base="agentType">
      <xsd:sequence>
        <xsd:element name="startExits" type="exitGroupType" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="endExits" type="exitGroupType" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="systemInfo" type="systemInfoType" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="transferSetType">
  <xsd:sequence>
    <xsd:element name="metaDataSet" type="metaDataSetType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="call" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="preSourceCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="postSourceCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="preDestinationCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="postDestinationCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="item" type="itemType"
maxOccurs="unbounded" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="index" type="xsd:nonNegativeInteger" use="optional" />
  <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="optional" />
  <xsd:attribute name="startTime" type="xsd:dateTime" use="required" />
  <xsd:attribute name="total" type="xsd:nonNegativeInteger" use="required" />
  <xsd:attribute name="bytesSent" type="xsd:nonNegativeInteger" use="required" />
</xsd:complexType>

<xsd:complexType name="itemType">
  <xsd:sequence>
    <xsd:element name="source" type="fileSourceChecksumType"
maxOccurs="1" minOccurs="1" />
    <xsd:element name="destination" type="fileDestinationChecksumType"
maxOccurs="1" minOccurs="1" />
    <xsd:element name="status" type="statusType"
maxOccurs="1" minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="mode" type="modeType" use="required" />
</xsd:complexType>

<xsd:complexType name="fileSourceChecksumType">
  <xsd:complexContent>
    <xsd:extension base="fileSourceType">
      <xsd:sequence>
        <xsd:element name="checksum" type="checksumType" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fileDestinationChecksumType">
  <xsd:complexContent>
    <xsd:extension base="fileDestinationType">
      <xsd:sequence>
        <xsd:element name="checksum" type="checksumType"
minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="actionType">
  <xsd:simpleContent>
    <xsd:extension base="actionEnumType">
      <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="actionEnumType">

```

```

    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="cancelled"/>
      <xsd:enumeration value="started"/>
      <xsd:enumeration value="progress"/>
      <xsd:enumeration value="completed"/>
      <xsd:enumeration value="malformed"/>
      <xsd:enumeration value="notAuthorized"/>
      <xsd:enumeration value="deleted"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="systemInfoType">
    <xsd:attribute name="architecture" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="version" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:element name="malformed">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="action" type="actionType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent" type="agentExitStatusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="status" type="statusType"
          maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
      <xsd:attribute name="agentRole" type="agentRoleType" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="notAuthorized">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="action" type="actionType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="originator" type="origRequestType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="authority" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
        <xsd:element name="status" type="statusType"
          maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
      <xsd:attribute name="agentRole" type="agentRoleType" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="statisticsType">
    <xsd:sequence>
      <xsd:element name="actualStartTime" type="xsd:dateTime"
        maxOccurs="1" minOccurs="0"/>
      <xsd:element name="retryCount" type="xsd:nonNegativeInteger"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="numFileFailures" type="xsd:nonNegativeInteger"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="numFileWarnings" type="xsd:nonNegativeInteger"
        maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="webGatewayType">
    <xsd:attribute name="webGatewayName" type="xsd:string" use="optional"/>
    <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional"/>
    <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="webUserType">
    <xsd:attribute name="webGatewayName" type="xsd:string" use="required"/>
    <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional"/>
    <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:schema>

```

Note: From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.

Understanding the transfer log message

<transaction>

Group element that specifies a group of transfers you want to perform together.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.
ID	Specifies the unique transaction ID. The ID can be a maximum of 48 alphanumeric characters.
relatedID	Optional. If the transaction is the delete or download of a file from a file space, relatedID specifies the transaction ID of the transfer that uploaded the file to the file space.
agentRole	Optional. Specifies whether the agent concerned is on the source or destination system
xmlns:xsi	Namespace declaration. Indicates that the elements and data types used in this schema derive from the "https://www.w3.org/2001/XMLSchema-instance" namespace.
xsi:noNamespaceSchemaLocation	Specifies the name and location of the XML schema document to validate this message against if there is no namespace declaration. The value you specify for this attribute must refer to a Managed File Transfer TransferLog.xsd document.

<action>

Describes the status of the file transfer at the time logged by the time attribute. The status can be one of the following values:

- started
- progress
- completed
- cancelled
- malformed (indicates the file transfer request message content can not be interpreted.)
- notAuthorized
- deleted

Attribute	Description
time	The time that the transfer status was captured, expressed in UTC format.

<sourceAgent>

Specifies the name of the agent on the system where the source file is located. Only <sourceAgent> or <sourceWebUser> can be specified.

<startExits>

Group element that contains one or more user exit elements. This element can occur once only.

<endExits>

Group element that contains one or more user exit elements. This element can occur once only.

<systemInfo>

Describes the system architecture, name, and version. This element can occur once only.

Attribute	Description
agent	The name of the agent on the source system.

Attribute	Description
QMgr	The name of the queue manager on the source system.
agentType	The type of the agent. Valid values are: <ul style="list-style-type: none"> • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

<destinationAgent>

Specifies the name of the agent on the system the file was transferred to. Either <destinationAgent> or <destinationWebUser> can be specified.

Attribute	Description
agent	The name of the agent on the destination system.
QMgr	The name of the queue manager on the destination system.
agentType	The type of the agent. Valid values are: <ul style="list-style-type: none"> • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

<startExits>

Group element that contains one or more user exit elements. This element can occur once only.

<endExits>

Group element that contains one or more user exit elements. This element can occur once only.

<systemInfo>

Describes the system architecture, name, and version. This element can occur once only.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

The IBM MQ user ID that was supplied in the message descriptor (MQMD)

<webUserID>

Optional. The user ID that was supplied to the web browser submitting the transfer request.

<webBrowser>

Optional. The web browser that the transfer request was submitted from.

<status>

The result code and supplement messages.

<trigger>

Group element that contains the trigger elements defined in the original transfer request. These elements can be either or both of the following:

<fileExist>

Trigger condition based on whether a file exists

<fileSize>

Trigger condition based on whether a file meets or exceeds the specified size

<transferSet>

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
startTime	Records the time that the set of transfers started, expressed in UTC format.
total	Specifies the total number of items in this set of transfers.
index	Optional attribute. Specifies the position of the first item in progress of the transfer set. The index attribute increments from zero. For example, if the index is set to 1, the progress message is the second of two items.
size	Optional attribute. Specifies the number of items in the progress report.
priority	Optional attribute. Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the source agent priority level.

<metaDataSet>

Group element containing one or more of the following attributes:

<metaData>

Attribute	Description
key	The key half of a metadata key-value pair. The <metaData> element content contains the value half of the pair. For example <metaData key="testkey1">testvalue1</metaData>

<job>

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: [“File transfer request message format” on page 755.](#)

<name>

The value of name can be any string.

<scheduleLog>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
ID	Matches the schedule ID if the transfer is a scheduled transfer.

<item>

Group element that contains elements specifying the source and destination file names and locations.

<source>

Group element that contains the <file> element or the <queue> element, and the <checksum> element for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: <ul style="list-style-type: none"> • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.
correlationBoolean	A boolean correlation value. If the source is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the source is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the source is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

<queue>

When used with the <source> element, specifies the name of the queue that the transferred messages were read from, which is located on the source agent queue manager.

Attribute	Description
messageCount	The number of messages that were read from the queue.
groupId	The IBM MQ group ID of the messages read from the queue.

<destination>

Group element that contains the <file> element or the <queue> element, and <checksum> element for the destination.

Only one of <file> and <queue> is present as a child element of destination.

Attribute	Description
type	<p>The type of destination. The valid options are as follows:</p> <ul style="list-style-type: none"> • queue - specifies an IBM MQ queue as the destination • file - specifies a file as the destination • directory - specifies a directory as the destination • <z/OS dataset> - specifies a z/OS data set as the destination • <z/OS pds> - specifies a z/OS partitioned data set as the destination <p>The option queue can be present only when the <destination> element has a child element of <queue>. The other options can be present only when the <destination> element has a child element of <file>.</p>
exist	<p>Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:</p> <ul style="list-style-type: none"> • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. <p>This attribute cannot be present if the <destination> element has a child element of <queue>.</p>
correlationBoolean	A boolean correlation value. If the destination is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the destination is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the destination is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

<file>

Specifies the absolute path of the file that was transferred (both at the source and destination). The fully-qualified path is in the format consistent with your operating system, for example C : / from / here . txt. File URIs are not used.

<queue>

When used with the <destination> element, specifies the name of the queue that was transferred to, which is located on any queue manager that is connected to the destination agent queue manager.

Attribute	Description
messageCount	The number of messages that were written to the queue.
messageLength	The length of the messages written to the queue.
groupId	If the transfer request specified that the file is split into multiple messages, the value of this attribute is the IBM MQ group ID of the messages written to the queue.
messageId	If the transfer request did not specify that the file is split into multiple messages, the value of this attribute is the IBM MQ message ID of the message written to the queue.

<checksum>

Optional element.

Specifies the type of hash algorithm that generated the message digest to create the digital signature. Currently Managed File Transfer supports Message Digest algorithm 5 (MD5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

<malformed>

Group element for malformed messages.

Attribute	Description
version	
ID	
agentRole	Either source agent or destination agent

<statistics>

Group element for statistical information for the transfer (when available).

<actualStartTime>

The actual time that the agent started running the transfer. Typically, the time is the same as (or very close to) the start time recorded for the transfer. However, when an agent is busy submitted transfers might be queued until the agent has capacity to run the transfers.

<retryCount>

The number of times that the transfer went into the recovery state and was retried by the agent. A transfer can go into a recovery state because the source and destination agents lose communication, either because of an IBM MQ network error or because they are not receiving data or acknowledgment messages for a period. This period is determined by the agent properties: transferAckTimeout and transferAckTimeoutRetries.

<numFileFailures>

The number of files in the transferSet that failed to transfer successfully.

<numFileWarnings>

The number of files in the transferSet that generated warnings while being transferred, but otherwise transferred successfully.

Examples

Examples of XML messages that conform to this schema are provided for each of the following types of transfer:

- [A transfer of a single file](#)
- [A transfer that contains multiple files](#)
- [A failed file transfer](#)
- [A transfer defined with a trigger](#)
- [A transfer started by a schedule](#)
- [A transfer that calls user exits](#)
- [A transfer through a Connect:Direct bridge node](#)

Single transfer log message examples

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/transfer_ID. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

Single file transfer - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent">
```

```

        xsi:noNamespaceSchemaLocation="TransferLog.xsd"
        xmlns="">
<action time="2011-01-26T13:03:26.484Z">started</action>
<sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</sourceAgent>
<destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
<originator>
  <hostName>dhcp-9-20-240-199.hursley.ibm.com.</hostName>
  <userID>mqm</userID>
  <mqmdUserID>mqm</mqmdUserID>
</originator>
<transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="0">
  <metaDataSet>
    <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingHost">dhcp-9-20-240-199.hursley.ibm.com.</
metaData>
  <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</
metaData>
  <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
  <metaData key="com.ibm.wmqfte.Priority">0</metaData>
  </metaDataSet>
</transferSet>
<scheduleLog ID="3"/>
</transaction>

```

Single file transfer success - progress

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:03:26.615Z">progress</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet index="0" size="1" startTime="2011-01-26T13:03:26.484Z" total="1"
bytesSent="1198">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="1151" last-modified="2009-11-02T10:37:01.000Z">/etc/passwd</file>
        <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
      </source>
      <destination type="file">
        <file size="1151" last-modified="2011-01-26T13:03:26.000Z">/tmp/passwd</file>
        <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>

```

Single file transfer success - completed

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:03:26.622Z">completed</action>

```

```

<sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</sourceAgent>
<destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</destinationAgent>
<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
  <mqmdUserID>mqm</mqmdUserID>
</originator>
<status resultCode="0">
  <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
</status>
<transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
  <metaDataSet>
    <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
    <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</
metaData>
    <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
  </metaDataSet>
</transferSet>
<statistics>
  <actualStartTime>2011-01-26T13:03:26.541Z</actualStartTime>
  <retryCount>0</retryCount>
  <numFileFailures>0</numFileFailures>
  <numFileWarnings>0</numFileWarnings>
</statistics>
</transaction>

```

Multiple file transfer log message examples

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/transfer_ID when a transfer that contains multiple files occurs.

Multiple file transfer - started

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:12:58.534Z">started</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="0">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
</transaction>

```

Multiple file transfer - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:12:58.753Z">progress</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet index="0" size="6" startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file01.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file02.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file02.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file03.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file03.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file04.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file04.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file05.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
```

```

file>      <file size="0" last-modified="2011-01-26T13:12:58.000Z"/>/srv/nfs/outgoing/file05.txt</
file>      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
          </destination>
          <status resultCode="0"/>
        </item>
        <item mode="binary">
          <source disposition="leave" type="file">
            <file size="0" last-modified="2011-01-26T13:10:19.000Z"/>/srv/nfs/incoming/file06.txt</
file>      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
          </source>
          <destination type="file">
            <file size="0" last-modified="2011-01-26T13:12:58.000Z"/>/srv/nfs/outgoing/file06.txt</
file>      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
          </destination>
          <status resultCode="0"/>
        </item>
      </transferSet>
</transaction>

```

Multiple file transfer - completed

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:12:58.766Z">completed</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</
metaData>
    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-01-26T13:12:58.634Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
  </statistics>
</transaction>

```

Failed file transfer log message examples

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_ID`. The XML examples show the log messages for a file transfer that fails being started, in progress, and completed.

File transfer failure - started

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"

```

```

        version="4.00"
        ID="414d51205553322e42494e44494e47538b0f404d03620020"
        agentRole="sourceAgent"
        xsi:noNamespaceSchemaLocation="TransferLog.xsd"
        xmlns=""
    <action time="2011-01-26T13:19:15.767Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</
metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>

```

File transfer failure - progress

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    version="4.00"
    ID="414d51205553322e42494e44494e47538b0f404d03620020"
    agentRole="sourceAgent"
    xsi:noNamespaceSchemaLocation="TransferLog.xsd"
    xmlns=""
    <action time="2011-01-26T13:19:15.944Z">progress</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet index="0" size="1" startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</
file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file>/srv/nfs/outgoing/file01.txt</file>
            </destination>
            <status resultCode="1">
                <supplement>BFGI00006E: File "/srv/nfs/outgoing/file01.txt" already exists.</
supplement>
            </status>
        </item>
    </transferSet>
</transaction>

```

File transfer failure - completed

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    version="4.00"
    ID="414d51205553322e42494e44494e47538b0f404d03620020"
    agentRole="sourceAgent"
    xsi:noNamespaceSchemaLocation="TransferLog.xsd"
    xmlns=""

```

```

<action time="2011-01-26T13:19:15.948Z">completed</action>
<sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</sourceAgent>
<destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</destinationAgent>
<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
  <mqmdUserID>mqm</mqmdUserID>
</originator>
<status resultCode="40">
  <supplement>BFGRP0034I: The file transfer request has
    completed with no files being transferred.
  </supplement>
</status>
<transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
  <metaDataSet>
    <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
    <metaData key="com.ibm.wmqfte.TransferId">414d5120555322e42494e44494e47538b0f404d03620020</
metaData>
    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
  </metaDataSet>
</transferSet>
<statistics>
  <actualStartTime>2011-01-26T13:19:15.878Z</actualStartTime>
  <retryCount>0</retryCount>
  <numFileFailures>1</numFileFailures>
  <numFileWarnings>0</numFileWarnings>
</statistics>
</transaction>

```

Triggered file transfer log message example

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_ID`. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

Trigger single file transfer success - started

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d31202020202020202020207e970d492000a102" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T22:05:18.703Z">started</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1"/>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1 </mqmdUserID>
  </originator>
  <trigger log="yes">
    <fileExist comparison="=" value="exist">c:\trigger.txt</fileExist>
  </trigger>
  <transferSet startTime="2008-11-02T22:05:18.703Z" total="1"></transferSet>
</transaction>

```


MFT user exit log message examples

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_ID`. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

Exit single file transfer proceed - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d3120202020202020207e970d492000d502" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="" >
  <action time="2008-11-02T22:36:13.046Z">started</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1"/>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1 </mqmdUserID>
  </originator>
  <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
    <metaDataSet>
      <metaData key="testkey1">testvalue1</metaData>
      <metaData key="testkey2">testvalue2</metaData>
    </metaDataSet>
  </transferSet>
</transaction>
```

Exit single file transfer proceed - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d3120202020202020207e970d492000d502"
  agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="" >
  <action time="2008-11-02T22:36:13.546Z">completed</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.SourceExit1">
        <status resultCode="proceed">
          <supplement>Source Start, modified metadata</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.SourceExit1">
        <status>
          <supplement>Source End</supplement>
        </status>
      </exit>
    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.DestinationExitProceed">
        <status resultCode="proceed">
          <supplement>Destination start, with proceed</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.DestinationExitProceed">
        <status>
          <supplement>destination end</supplement>
        </status>
      </exit>
    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
```

```

        version="6.1 build 7601 Service Pack 1"/>
</destinationAgent>
<originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1      </mqmdUserID>
</originator>
<transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
  <metaDataSet>
    <metaData key="newkey2">newvalue2</metaData>
    <metaData key="newkey1">newvalue1</metaData>
    <metaData key="newkey4">newvalue4</metaData>
    <metaData key="newkey3">newvalue3</metaData>
    <metaData key="newkey5">newvalue5</metaData>
    <metaData key="testkey1">testvalue1</metaData>
    <metaData key="testkey2">testvalue2</metaData>
  </metaDataSet>
</transferSet>
</transaction>

<!--
  In this example the source transfer start exit has modified the
  metadata as follows:

  Added keys and values for:
  newkey1, newvalue1
  newkey2, newvalue2
  newkey3, newvalue3
  newkey4, newvalue4
  newkey5, newvalue5

  Replaced values for:
  key1 to modifiedValue1

  Deleted keys and values for:
  key2
-->

```

Exit single file transfer cancel - canceled

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d31202020202020202020207e970d492000c702" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T22:25:59.328Z">cancelled</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.SourceExit1">
        <status resultCode="proceed">
          <supplement>Source Start, modified metadata</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.SourceExit1">
        <status>
          <supplement>Source End</supplement>
        </status>
      </exit>
    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.DestinationExit1">
        <status resultCode="cancelTransfer">
          <supplement>Destination start, with cancel</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.DestinationExit1">
        <status>
          <supplement>destination end</supplement>
        </status>
      </exit>
    </endExits>
  </destinationAgent>

```

```

    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1      </mqmdUserID>
  </originator>
  <transferSet startTime="2008-11-02T22:25:59.078Z" total="1"/>
</transaction>

```

Connect:Direct bridge transfer log message examples

The destinationAgent or sourceAgent element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

Source agent is Connect:Direct bridge agent Started:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d20092507"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T13:05:01.838Z">started</action>
  <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE" bridgeNode="CDNODE_VARUNA">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_KUIPER" agent="IXION"/>
  <originator>
    <hostName>kuiper.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="0" startTime="2011-03-07T13:05:01.838Z" total="1">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
</transaction>

```

Progress:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d20092507"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T13:05:03.448Z">progress</action>
  <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VARUNA" pnode="CDNODE_VARUNA" snode="CDNODE_ERIS">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>kuiper.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T13:05:01.838Z" total="1">

```

```

    <item mode="binary">
      <source disposition="leave" processName="f2007567" processNumber="68" type="file">
        <file last-modified="2011-03-07T13:05:02.573Z" size="4">CDNODE_ERIS:D:/AGENTS/
CDNODE_ERIS/test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </source>
      <destination type="file">
        <file last-modified="2011-03-07T13:05:03.338Z" size="4">D:\AGENTS\IXION\test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>

```

Completed:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d20092507"
  agentRole="sourceAgent"
  version="4.00" xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T13:05:03.495Z">completed</action>
  <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE">
    bridgeNode="CDNODE_VARUNA" pNode="CDNODE_VARUNA" sNode="CDNODE_ERIS">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>kuiper.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet bytesSent="48" startTime="2011-03-07T13:05:01.838Z" total="1">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-03-07T13:05:02.041Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
  </statistics>
</transaction>

```

Destination agent is Connect:Direct bridge agent Started:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:44.854Z">started</action>
  <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_ASTEROID" agent="VESTA"/>
  <originator>
    <hostName>belt.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>

```

```

</originator>
<transferSet bytesSent="0" startTime="2011-03-07T10:29:44.854Z" total="1">
  <metaDataSet>
    <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
    <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
    <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
    <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</
metaData>
    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
  </metaDataSet>
</transferSet>
</transaction>

```

Progress:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:46.682Z">progress</action>
  <sourceAgent QMgr="QM_asteroid" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_asteroid" agent="VESTA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VESTA" pNode="CDNODE_VESTA" sNode="CDNODE_HYGIEA">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>belt.example.com</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T10:29:44.854Z" total="1">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file last-modified="2011-03-04T14:53:28.323Z" size="4">D:\AGENTS\PALLAS\test.txt</
file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </source>
      <destination processName="f2006965" processNumber="59" type="file">
        <file size="4">CDNODE_VESTA:D:/AGENTS/CDNODE_VESTA/test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>

```

Completed:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:46.698Z">completed</action>
  <sourceAgent QMgr="QM_asteroid" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_asteroid" agent="VESTA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VESTA" pNode="CDNODE_VESTA" sNode="CDNODE_HYGIEA">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>belt.example.com</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet bytesSent="48" startTime="2011-03-07T10:29:44.854Z" total="1">
    <metaDataSet>

```

```

        <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
        <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
        <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
        <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
        <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
        <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</
metaData>
        <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
</transferSet>
<statistics>
    <actualStartTime>2011-03-07T10:29:45.010Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
</statistics>
</transaction>

```

Scheduled file transfer log message formats

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its `SYSTEM.FTE/Log/agent name/schedule ID` topic). This message conforms to the `ScheduleLog.xsd` XML schema.

Schema

The following schema describes which elements are valid in a schedule log XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>

  <xsd:element name="schedulelog">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="hostUserIDType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="action" type="actionType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="schedule" type="scheduleType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent" type="agentType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationAgent" type="agentClientType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="status" type="statusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="transferSetType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="job" type="jobType"
          maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="actionType">
    <xsd:simpleContent>
      <xsd:extension base="actionEnumType">
        <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="actionEnumType">
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="submit"/>
      <xsd:enumeration value="delete"/>
      <xsd:enumeration value="expire"/>
      <xsd:enumeration value="skipped"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="transferSetType">
    <xsd:sequence>
      <xsd:element name="item" type="itemType"
        maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

```

```

    </xsd:sequence>
    <xsd:attribute name="size" type="xsd:int" use="required"/>
    <xsd:attribute name="priority" type="priorityType" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="itemType">
    <xsd:sequence>
      <xsd:element name="source" type="fileSourceType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="destination" type="fileDestinationType"
        maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required"/>
    <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required"/>
  </xsd:complexType>
</xsd:schema>

```

Understanding the schedule log message

The elements and attributes used in the schedule log message are described:

<schedulelog>

Group element that describes a single submitted scheduled file transfer.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.
ID	The unique identifier for the submitted schedule file transfer.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

The MQ user ID that was supplied in the message descriptor (MQMD)

<action>

Specifies the action to take with the scheduled transfer matching the ID attribute of <schedulelog> element. This element must be one of the following values:

- submit - new scheduled transfer
- delete - cancel schedule transfer
- expire - schedule transfer entry about to be processed
- skipped - a transfer that was scheduled cannot be started because the agent is offline. This message is logged when the agent becomes available to indicate the transfer was skipped.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<status>

The result code and supplement messages.

<transferSet>

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
size	Specifies the number of transfer items.
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

<item>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as being either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. Permitted values are MD5 or none

<source>

Group element that contains the <file> and <checksum> elements for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: <ul style="list-style-type: none"> • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

<destination>

Group element that contains the <file> and <checksum> elements for the file on the destination system.

Attribute	Description
type	The type of file or directory at the destination. The valid options are as follows: <ul style="list-style-type: none"> • file - specifies a file as the destination • directory - specifies a directory as the destination •  dataset - specifies a z/OS data set as the destination

Attribute	Description
	<ul style="list-style-type: none"> • z/OS PDS - specifies a z/OS partitioned data set as the destination
exist	<p>Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:</p> <ul style="list-style-type: none"> • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file.

<file>

Specifies the name of the file to transfer. Use the fully qualified path in the format consistent with your operating system, for example C : / from / here . txt. Do not use file URIs.

Attribute	Description
encoding	The encoding for a text file transfer.
EOL	<p>Specifies the end of line marker. Permitted values are:</p> <ul style="list-style-type: none"> • LF - line feed character only • CRLF - carriage return and line feed character sequence

<job>

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: [“File transfer request message format” on page 755.](#)

<name>

The value of name can be any string.

Examples

Examples of XML messages that conform to this schema are provided for each of the following scheduled transfer actions:

- [A scheduled transfer is created](#)
- [A scheduled transfer is canceled](#)
- [A schedule transfer expires](#)

Transfers that are started by a schedule are logged in the same way as a standard transfer. For examples of log messages for transfers started by a schedule, see [“Scheduled file transfer log message examples” on page 738.](#)

Schedule file transfer log message examples

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/schedule_ID when a scheduled transfer action occurs.

Scheduled transfer log message

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/agent name/schedule ID topic). This message conforms to the ScheduleLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
```

```

<originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
</originator>
<action time="2008-11-23T21:32:01Z">submit</action>
<schedule>
  <submit timebase="admin" timezone="Europe/London">2008-11-23T22:00</submit>
</schedule>
<sourceAgent agent="FTEAGENT" QMgr="QM1"/>
<destinationAgent agent="FTEAGENT" QMgr="QM1"/>
<status resultCode="0"/>
<transferSet size="1" priority="0">
  <item mode="binary" checksumMethod="MD5">
    <source recursive="false" disposition="leave">
      <file>c:\sourcefiles\source1.doc</file>
    </source>
    <destination type="file" exist="overwrite">
      <file>c:\destinationfiles\dest1.doc</file>
    </destination>
  </item>
</transferSet>
</schedulelog>

```

This message is a log of the following information:

- Who originated the request
- When the request was submitted
- When the scheduled transfer starts
- The source and destination agent details
- The transfer specification

The ID attribute of the <schedulelog> element is a unique ID for this scheduled transfer (in the source agent). This ID is used to correlate schedule entries with the actual file transfers.

The <action> element value of submit confirms the request has been received.

Scheduled transfer cancel log message

When a request to cancel a pending scheduled file transfer is received by the agent, the following message is published to the SYSTEM.FTE/Log/agent_name topic:

```

<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <action time="2008-11-23T21:56:27Z">delete</action>
  <status resultCode="0"/>
</schedulelog>

```

The ID attribute value corresponds to the ID of the pending transfer request ID in the schedules message.

Scheduled transfer expire log message

When the current time matches the time of the earliest pending file transfer in the schedule list (as indicated by the value of the <next> element), a schedule log message is published to indicate that the scheduled transfer entry has expired:

```

<?xml version="1.0" encoding="UTF-8"?>
<schedulelog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00" ID="3"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>

```

```

</originator>
<action time="2011-01-26T13:03:26Z">expire</action>
<sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
<status resultCode="0"/>
</schedulelog>

```

The <action> element value of "expire" confirms the schedule entry has now been removed from the schedule list and is being processed. A schedule message for the agent is published with the expired entry no longer present.

Related reference

[“Scheduled file transfer log message formats” on page 744](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/agent name/schedule ID topic). This message conforms to the ScheduleLog.xsd XML schema.

[“Scheduled file transfer log message examples” on page 738](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/transfer_ID. The XML examples show the log messages that are created when a file transfer occurs as a result of a schedule.

MFT monitor log message format

Monitor log messages are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/Monitors/monitor_name/monitor_ID.

If you want to collect data or view monitor actions, set up a subscription to a wildcard topic tailored to the monitors that you are interested in. For example:

```
Log/#
```

or,

```
Log/agent_name/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

The MonitorLog.xsd schema document is located in the MQ_INSTALLATION_PATH/mqft/samples/schema directory. The MonitorLog.xsd schema imports fteutils.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor log XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="monitorLog">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="hostUserIDType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="references" type="referencesType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="action" type="monitorActionType" maxOccurs="1"
minOccurs="1" />
        <xsd:element name="monitorAgent" type="agentType" maxOccurs="1"
minOccurs="1" />
        <xsd:element name="status" type="statusType" maxOccurs="1"
minOccurs="1" />
        <xsd:element name="monitorMetaData" type="monitorMetaDataType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="monitorExits" type="exitGroupType" maxOccurs="1"
minOccurs="0" />

```

```

    <xsd:element name="jobDetails" type="jobType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="taskXMLRequest" type="taskXMLRequestType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="monitorXMLRequest" type="monitorXMLRequestType"
maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required"/>
    <xsd:attribute name="monitorName" type="xsd:string" use="required"/>
    <xsd:attribute name="referenceId" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="monitorActionType">
  <xsd:simpleContent>
    <xsd:extension base="monitorActionEnumType">
      <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="monitorActionEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="create"/>
    <xsd:enumeration value="delete"/>
    <xsd:enumeration value="start"/>
    <xsd:enumeration value="stop"/>
    <xsd:enumeration value="triggerSatisfied"/>
    <xsd:enumeration value="triggerNotSatisfied"/>
    <xsd:enumeration value="triggerFail"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="monitorMetaDataType">
  <xsd:sequence>
    <xsd:element name="originalMetaData" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="updatedMetaData" type="metaDataSetType" maxOccurs="unbounded"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="taskXMLRequestType">
  <xsd:sequence>
    <xsd:element name="originalRequest" type="xsd:string" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="updatedRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="taskId" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="referencesType">
  <xsd:sequence>
    <xsd:element name="createRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="taskRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorXMLRequestType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="type" type="xmlContentEnumType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="xmlContentEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="escapedXML"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Understanding the monitor log message

The elements and attributes used in the monitor log messages are described in the following list:

<monitorLog>

Group element containing the elements describe an action that has been performed by a monitor.

Attribute	Description
version	Required. The version of the monitor list message format.
monitorName	Required. The name of the monitor. Unique for the agent that the monitor is defined on.
referenceId	The ID of the monitor action.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<references>

References to the IDs of other messages associated with this monitor action.

<createRequest>

The message ID of the XML request message that was used to create the monitor.

<taskRequest>

The message ID of the XML request message that the monitor submits as a result of this action.

<action>

The action that occurred, which this log message is associated with. The value inside the element can be one of the following: create, delete, start, stop, triggerSatisfied, triggerNotSatisfied, or triggerFail.

<monitorAgent>

The agent that is monitoring the resource.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The name of the queue manager that the agent connects to.
bridgeURL	Optional. If the agent is a protocol bridge agent, the URL of the protocol server.

<status>

The status of the resource monitor action being logged.

Attribute	Description
resultCode	Required. The integer result code from the action.

<supplement>

Additional information about the status of the resource monitor action being logged.

<monitorMetaData>

Group element that contains the <originalMetaData> and <updatedMetaData> elements.

<originalMetaData>

Element that contains one or more <metadata> elements that describe the metadata of the monitor before the action occurs.

<updatedMetaData>

Element that contains one or more <metadata> elements that describe the metadata of the monitor after the action occurs.

<metadata>

Defines a metadata key-value pair. The key is an attribute of the element; the value is the content of the element.

Attribute	Description
key	The key of the metadata.

<monitorExits>

Group element containing one or more <exit> elements.

<exits>

Element describing an exit run by the resource monitor.

Attribute	Description
name	Required. The name of the resource monitor exit.

<status>

The status of the resource monitor exit that is being logged.

Attribute	Description
resultCode	Required. The integer result code from the exit.

<supplement>

Additional information about the status of the resource monitor exit that is being logged.

<jobDetails>

Element containing a single <name> element.

<name>

The name of the job..

<taskXMLRequest>

Group element that contains the <originalRequest> and <updatedRequest> elements.

Attribute	Description
taskId	The ID of the task request message.

<originalRequest>

Element that contains the escaped XML request message for the task that the monitor performs.

<updatedRequest>

Element that contains the updated escaped XML request message for the task that the monitor performs.

<monitorXMLRequest>

The monitor XML request.

Attribute	Description
type	Required. The format of the monitor XML request data inside of the <monitorXMLRequest> element. The only valid value is escapedXML.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor actions:

- [A monitor is created](#)
- [The condition of a monitor is satisfied when the monitor polls the resource](#)

- The condition of a monitor is not satisfied when the monitor polls the resource
- A monitor is deleted

Related reference

[“MFT monitor log message examples” on page 753](#)

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/monitor_ID when a monitor action occurs.

MFT monitor log message examples

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent_name/monitor_ID when a monitor action occurs.

Monitor created log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORTWO"
  referenceId="414d51205553322e42494e44494e47538b0f404d04410020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d51205553322e42494e44494e47538b0f404d04410020</createRequest>
  </references>
  <action time="2011-01-26T12:41:24Z">start</action>
  <monitorAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER" />
  <status resultCode="0"/>
</monitorLog>
```

Monitor condition satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORONE"
  referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
  </references>
  <action time="2011-01-26T12:56:46Z">triggerSatisfied</action>
  <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS" />
  <status resultCode="0"/>
  <monitorMetaData>
    <originalMetaData>
      <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
      <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
      <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
      <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
      <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
      <metaData key="FILENAME">new.completed</metaData>
      <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
      <metaData key="LASTMODIFIEDTIME">12.56</metaData>
      <metaData key="FILESIZE">0</metaData>
      <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
    </originalMetaData>
    <updatedMetaData>
      <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
      <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
      <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
      <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
      <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
      <metaData key="FILENAME">new.completed</metaData>
      <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
      <metaData key="LASTMODIFIEDTIME">12.56</metaData>
    </updatedMetaData>
  </monitorMetaData>
</monitorLog>
```

```

    <metaData key="FILESIZE">0</metaData>
    <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
  </updatedMetaData>
</monitorMetaData>
<taskXMLRequest taskId="null">
  <originalRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
      &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
      &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
      &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
      &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
      &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
        &lt;source disposition="leave" recursive="false"&gt;
          &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
          &lt;destination exist="error" type="directory"&gt;
            &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
          &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
        &lt;/originalRequest>
      <updatedRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
          &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
          &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
          &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
          &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
          &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
            &lt;source disposition="leave" recursive="false"&gt;
              &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;
              &lt;destination exist="error" type="directory"&gt;
                &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
              &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
            &lt;/updatedRequest>
          </taskXMLRequest>
        </monitorLog>

```

Monitor condition not satisfied log message

```

<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORONE"
  referenceId="414d5120553322e42494e44494e47538b0f404d09430020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d5120553322e42494e44494e47538b0f404d09430020</createRequest>
  </references>
  <action time="2011-01-26T12:58:46Z">triggerNotSatisfied</action>
  <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <status resultCode="0"/>
</monitorLog>

```

Monitor deleted log message

```

<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:lst="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  agent="AGENT_JUPITER"
  monitor="MONITORONE"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition
MonitorList.xsd">
  <status state="deleted"/>
  <configuration>
    <description/>
    <resources>
      <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
      <conditions>
        <condition>
          <name/>

```

```

        <resource id="" />
        <fileMatch>
            <pattern>*.completed</pattern>
        </fileMatch>
    </condition>
</conditions>
</triggerMatch>
<tasks>
    <task>
        <name/>
        <description/>
        <taskXML>&lt;&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;&lt;request
            xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
            xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;&lt;managedTransfer&gt;
                &lt;&lt;originator&gt;&lt;&lt;hostName&gt;example.ibm.com.&lt;&lt;/hostName&gt;
                &lt;&lt;userID&gt;mqm&lt;&lt;/userID&gt;&lt;&lt;/originator&gt;
                &lt;&lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                &lt;&lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                &lt;&lt;transferSet&gt;&lt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                &lt;&lt;source disposition="leave" recursive="false"&gt;
                &lt;&lt;file&gt;/srv/nfs/incoming/*.txt&lt;&lt;/file&gt;&lt;&lt;/source&gt;
                &lt;&lt;destination exist="error" type="directory"&gt;
                &lt;&lt;file&gt;/srv/backup&lt;&lt;/file&gt;&lt;&lt;/destination&gt;
                &lt;&lt;/item&gt;&lt;&lt;/transferSet&gt;&lt;&lt;/managedTransfer&gt;&lt;&lt;/request&gt;
            &lt;&lt;/taskXML>
        </task>
    </tasks>
</configuration>
<pollInterval units="minutes">1</pollInterval>
<batch maxSize="1"/>
</lst:monitorList>

```

File transfer request message format

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

File transfer messages can have one of following three root elements:

- `<request>` - for new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
- `<cancel>` - for canceling file transfers in progress
- `<transferSpecifications>` - for specifying multiple transfer file groups, used by the **fteCreateTransfer** command

For information about specifying multiple transfer groups by using the `<transferSpecifications>` element, see [Using transfer definition files](#).

Schema

The following schema describes which elements are valid in a transfer request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
<xsd:include schemaLocation="fteutils.xsd"/>
<!--
    Defines the request of a managed transfer and version number
    <request version="1.00" ...
        <managedTransfer>
            ...
        </managedTransfer>
    </request>
-->
<xsd:element name="request">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="managedTransfer" type="managedTransferType"/>
            <xsd:element name="deleteScheduledTransfer" type="deleteScheduledTransferType"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>

```

```

        <xsd:element name="managedCall" type="managedCallType"/>
    </xsd:choice>
    <xsd:attribute name="version" type="versionType" use="required"/>
</xsd:complexType>
</xsd:element>

<!--
  Defines the cancel request of a managed transfer and version number
  <cancel version="1.00"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <originator>
      <hostName>myMachine</hostName>
      <userID>myUserId</userID>
    </originator>      - Delete a scheduled transfer.

    <transfer>
      Transfer ID to Cancel
    </transfer>
  </cancel>
-->
<xsd:element name="cancel">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="originator" type="hostUserIDType" maxOccurs="1" minOccurs="1"/>
      <xsd:choice>
        <xsd:element name="transfer" type="IDType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="call" type="IDType" maxOccurs="1" minOccurs="1"/>
      </xsd:choice>
      <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required"/>
  </xsd:complexType>
</xsd:element>

<!--
  Defines the transfer definition element structure.
  <transferSpecifications>
    <item ...
    <item ...
  </transferSpecifications>
-->
<xsd:element name="transferSpecifications">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="item" type="itemType" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!--
  Define a managed transfer of an instigator and request
  <managedTransfer>

    <originator>
      ...
    </originator>

    <schedule>
      <submit timebase="source"|"UTC">2008-12-07T16:07</submit>
      <repeat>
        <frequency interval="hours">2</frequency>
        <expireTime>2008-12-0816:07</expireTime>
      </repeat>
    </schedule>

    <sourceAgent agent="here" QMgr="near"/>
    <destinationAgent agent="there" QMgr="far"/>

    <trigger>
      ...
    </trigger>

    <transferSet>
      ...
    </transferSet>
  </managedTransfer>
-->

<xsd:complexType name="managedTransferType">
  <xsd:sequence>
    <xsd:element name="originator" type="origTransferRequestType" maxOccurs="1"

```

```

minOccurs="1"/>
  <xsd:element name="schedule" type="scheduleType" maxOccurs="1" minOccurs="0"/>
  <xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
  <xsd:element name="destinationAgent" type="agentClientType" maxOccurs="1" minOccurs="1"/>
  <xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0"/>
  <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
  <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
  <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>

<!--
  This is a modified form of origRequestType which is used on a managed transfer request.
  The hostName and userID are mandatory attributes in this case.
-->
<xsd:complexType name="origTransferRequestType">
  <xsd:sequence>
    <xsd:element name="hostName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="userID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Defines the transferset of source and destination agent and one or more files
  <transferset priority="1">
    <metaDataSet>
      <metaData key="keyname">keyvalue</metaData>
      <metaData key="keyname">keyvalue</metaData>
    </metaDataSet>

    <item>
      . . .
    </item>
  </transferset>
-->
<xsd:complexType name="transferSetType">
  <xsd:sequence>
    <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="preSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="postSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="preDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="postDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="item" type="itemType" maxOccurs="unbounded" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="priority" type="priorityType" use="optional"/>
</xsd:complexType>

<!--
  Define a file pair with source and destination
  <item mode=[binary|text]>
    <source recursive="false" disposition="leave">
      <file>filename</file>
    </source>

    <destination type="file" exist="error">
      <file>filename</file>
    </destination>

  </item>
-->
<xsd:complexType name="itemType">
  <xsd:sequence>
    <xsd:element name="source" type="fileSourceType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="destination" type="fileDestinationType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="mode" type="modeType" use="required"/>
  <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required"/>
</xsd:complexType>

<!--
  Defines the request to delete scheduled file transfer.
  <deleteScheduledTransfer>
    <originator>
      <delete>

```

```

        <hostName>myMachine</hostName>
        <userID>myUserId</userID>
    </delete>
</originator>
<ID>56</ID>
</deleteScheduledTransfer>
-->
<xsd:complexType name="deleteScheduledTransferType">
    <xsd:sequence>
        <xsd:element name="originator" type="origDeleteType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="ID" type="idType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="managedCallType">
    <xsd:sequence>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="callTransferSetType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="callTransferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="call" type="commandActionType" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional"/>
</xsd:complexType>
</xsd:schema>

```

Understanding the transfer request message

The elements and attributes used in transfer request messages are described in the following list:

Element descriptions

<request>

Group element containing all the elements required to specify a file transfer request.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<managedTransfer>

Group element that contains all the elements required for a single file transfer or single group of file transfers.

<deleteScheduledTransfer>

Group element that contains originator and ID information to cancel a schedule transfer.

<managedCall>

Group element that contains all the elements required for a single managed call of a program or executable.

<ID>

Unique identifier that specifies the transfer request to delete from the list of pending scheduled transfers.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<schedule>

Group element describing the scheduled time for the file transfer, the repeat behavior, and when the next occurrence is due.

<submit>

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. This attribute can have one of the following values: <ul style="list-style-type: none"> • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time
timezone	The time zone description according to the timebase value

<repeat>

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

<frequency>

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> • minutes • hours • days • weeks • months • years

<expireTime>

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

<expireCount>

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.
hostName	The host name or IP address of the agent queue manager.
portNumber	The port number used for client connections to the destination agent queue manager.
channel	The channel name used to connect to the destination agent queue manager.

<trigger>

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The valid values are as follows: <ul style="list-style-type: none"> • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

<fileExist>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileSize> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid values are as follows: <ul style="list-style-type: none"> • = at least one file name in the name list must match • != a minimum of one of the files in the name list does not exist
value	Indicates the comparison type: <ul style="list-style-type: none"> • exist: file must exist

<fileSize>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileExist> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid value is as follows: <ul style="list-style-type: none"> • >= one of the file names in the name list exists and has a minimum size as specified in the value attribute
value	File size specified as an integer value with units specified as one of the following: <ul style="list-style-type: none"> • B - bytes • KB - kilobytes • MB - megabytes

Attribute	Description
	<ul style="list-style-type: none"> GB - gigabytes (the units value is not case-sensitive)

<reply>

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file or the default of `WMQFTE.*` if not specified.

Attribute	Description
detailed	Whether detailed transfer result information is required in the reply message. Multiple reply messages for each transfer can be generated. The valid values are as follows: <ul style="list-style-type: none"> true - detailed reply information is required. The format of the information is the same as that published to the transfer log in the progress messages, that is, the <code><transferSet></code> element. For more information, see “File transfer log message formats” on page 723. Detailed reply information is present only when the transfer source agent has the <code>enableDetailedReplyMessages</code> property set to true. false - detailed reply information is not required. The default value is false.
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.
persistent	Whether the message written to the reply queue is persistent. The valid values are as follows: <ul style="list-style-type: none"> true - the message is persistent false - the message is not persistent qdef - the persistence of the message is defined by the properties of the reply queue The default value is false.

<transferSet>

Specifies a group of file transfers you want to perform together or a group of managed calls that you want to perform together. During transmission `<transferSet>` is a group element containing `<item>` elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

<metaDataSet>

Optional group element containing one or more metadata items.

<metaData>

Specifies the user-defined metadata that is passed to the exit points called by the agent. The element contains the metadata value as a string.

Attribute	Description
key	Metadata name as a string

<call>

Group element that contains <command> elements specifying the program or executable to call.

<command>

Specifies the program or executable to call. The command must be located on the agent command path. For more information, see [Table 45 on page 528](#). This element can contain optional <argument> elements.

Attribute	Description
name	The name of the command.
successRC	The successful return code that this command returns. Default is 0.
retryCount	The number of times that the command is to be retried if it fails.
retryWait	The time, in seconds, to wait between retries of the command.
type	The type of program to be called. The valid values are antscript, jcl, or executable.

<argument>

Specifies an argument to pass to the command.

<item>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. The valid values are MD5 or none.

<source>

Group element that specifies files on the source system and whether they are removed after the transfer completes

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid values are as follows: <ul style="list-style-type: none"> • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

<file>

Specifies the transfer source. **Multi** For Multiplatforms, the transfer source can be a file or a directory name. For the z/OS platform, the transfer source can be a file, directory, data set, or PDS name. Use the fully qualified path in the format consistent with your operating system, for example C:/from/here.txt. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the source file. This alias is the name of the source file, excluding any directory path specified for the transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are:

Attribute	Description
	<ul style="list-style-type: none"> • LF - line feed character only • CRLF - carriage return and line feed character sequence
encoding	The encoding of the source file for a text file transfer.
 delimiter	Specifies the delimiter that is included between records in record-oriented source files, for example, z/OS data sets. Specify the delimiter value as two hexadecimal digits in the range 00-FF, prefixed by x. For example, x12 or x03,x7F.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values is as follows: <ul style="list-style-type: none"> • binary - a hexadecimal delimiter This attribute is available only if you have enabled the 7.0.4.1 function.
delimiterPosition	Specifies the position to insert delimiters when writing record-oriented source file records to a normal file. The valid values are as follows: <ul style="list-style-type: none"> • prefix - the delimiter is inserted into the destination file before the data from each source record-oriented file record. • postfix - the delimiter is inserted into the destination file after the data from each source record-oriented file record.
includeDelimiterInFile	Specifies whether to include a delimiter between records in record-oriented source files.
 keepTrailingSpaces	Specifies whether trailing spaces are to be kept on source records read from a fixed-length-format data set as part of a text mode transfer. The default is that trailing spaces are stripped. The valid values are as follows: <ul style="list-style-type: none"> • true - trailing spaces are kept on source records read from a fixed-length-format data set • false - trailing spaces are stripped from source records read from a fixed-length-format data set

<queue>

When used with the <source> element, specifies the name of the queue to transfer from, which must be located on the source agent queue manager. Use the format *QUEUE*. Do not include the queue manager name, the queue must be present on the source agent queue manager. You cannot use the <queue> element inside the <source> element, if you have used it inside of the <destination> element.

Attribute	Description
useGroups	Specifies whether to transfer only the first complete group of messages from the source queue. The valid values are as follows: <ul style="list-style-type: none"> • true - transfer only the first complete group of messages • false - transfer all messages on the source queue
groupId	Specifies the group of messages to read from the source queue. This attribute is valid only when the value of the useGroups attribute is true.

Attribute	Description
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values are as follows: <ul style="list-style-type: none"> • text - a text or Java literal delimiter • binary - a hexadecimal delimiter
delimiter	Specifies the delimiter that is included in the destination file between individual message data.
delimiterPosition	Specifies whether the delimiter is included in the destination file before or after individual message data. The valid values are as follows: <ul style="list-style-type: none"> • prefix - the delimiter is included before the data • postfix - the delimiter is included after the data
encoding	Specifies the source queue encoding.
waitTime	Specifies the time, in seconds, for the source agent to wait for either: <ul style="list-style-type: none"> • a message to appear on the source queue, if the queue is empty or has become empty • a complete group to appear on the source queue, if the useGroups attribute has been set to true <p>For information about setting the waitTime value, see “Guidance for specifying a wait time on a message-to-file transfer” on page 657.</p>

<destination>

Group element that specifies the destination and the behavior if files exist at the destination agent.

You can specify only one of <file> and <queue> as a child element of destination.

Attribute	Description
type	The type of destination. The valid values are as follows: <ul style="list-style-type: none"> • file - specifies a file as the destination • directory - specifies a directory as the destination •  dataset - specifies a z/OS data set as the destination •  pds - specifies a z/OS partitioned data set as the destination • queue - specifies an IBM MQ queue as the destination • filespace - specifies a file space as the destination <p>The value queue is valid only when the <destination> element has a child element of <queue>.</p> <p>The value filespace is valid only when the <destination> element has a child element of <filespace>.</p> <p>The other values are valid only when the <destination> element has a child element of <file>.</p>
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid values are as follows:

Attribute	Description
	<ul style="list-style-type: none"> error - reports an error and the file is not transferred. overwrite - overwrites the existing destination file. <p>This attribute is not valid if the <destination> element has a child element of <queue> or <filesystem>.</p>

<file>

Specifies additional settings for the previously-described **<destination>** element. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the <destination> file. This alias is the name of the source file, excluding any directory path specified for the transfer.
encoding	The encoding of the <destination> file for a text file transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: <ul style="list-style-type: none"> LF - line feed character only CRLF - carriage return and line feed character sequence
truncateRecords	Optional. Specifies that <destination> records longer than the LRECL data set attribute are truncated. <ul style="list-style-type: none"> True - the records are truncated False - the records are wrapped <p>The default setting is false.</p>

<queue>

When used with the **<destination>** element, specifies the name of the queue to transfer to, which can be located on any queue manager that is connected to the destination agent queue manager. Use the format `QUEUE@QM` where `QUEUE` is the name of the queue to put the messages on and `QM` is the queue manager where the queue is located. You cannot use the **<queue>** element inside the **<destination>** element, if you have used it inside of the **<source>** element.

Attribute	Description
delimiter	The delimiter to split the file into multiple messages.
delimiterType	Specifies the type of delimiter. The valid values are as follows: <ul style="list-style-type: none"> text - a Java regular expression binary - a sequence of hexadecimal bytes size - a number of bytes, kibibytes, or mebibytes. For example, 1 B, 1 K, or 1 M.
delimiterPosition	Specifies whether the delimiter is expected before or after the data to include in individual messages. The valid options are as follows: <ul style="list-style-type: none"> prefix - the delimiter is expected before the data postfix - the delimiter is expected after the data
includeDelimiterInMessage	A boolean specifying whether to include the delimiters that were used to split the file into multiple messages at the end of the messages.

Attribute	Description
encoding	Specifies the destination queue encoding.
persistent	Specifies whether the messages are persistent. The valid values are as follows: <ul style="list-style-type: none"> • true - the messages are persistent • false - the messages are not persistent • qdef - the persistence value of the messages is defined by the settings on the destination queue
setMqProps	A boolean specifying whether IBM MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
unrecognisedCodePage	Specifies whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager. The valid values are as follows: <ul style="list-style-type: none"> • fail - the transfer reports a failure • binary - the data is converted to the destination code page and the IBM MQ message header describing the format of the data is set to MQFMT_NONE. <p>The default behavior is fail.</p>

<filespace>

Group element specifying the name of the file space to transfer to.

<name>

When used with the <filespace> element, the value of this element specifies the name of the file space.

<preSourceCall>

Group element specifying a command to call at the source of the transfer, before the transfer starts.

<postSourceCall>

Group element specifying a command to call at the source of the transfer, after the transfer completes.

<preDestinationCall>

Group element specifying a command to call at the destination of the transfer, before the transfer starts.

<postDestinationCall>

Group element specifying a command to call at the destination of the transfer, after the transfer completes.

<command>

When used with the <preSourceCall>, <postSourceCall>, <preDestinationCall>, or <postDestinationCall> element, this element specifies the command to be called. The command must be located on the agent command path. For more information, see [Table 45 on page 528](#).

Attribute	Description
name	The name of the command to run.
successRC	The return code that is expected if the command runs successfully.

<argument>

When used with the <command> element, this element specifies an argument to be passed in to the command. You can have any number of <argument> elements inside a <command> element.

<job>

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: [“File transfer log message formats” on page 723.](#)

<name>

When used with the <job> element, the value of this element specifies the name of the job.

<transferSpecifications>

Group element that contains <item> elements for multiple transfer groups. See [Using transfer definition files](#) for further details about how to use this element.

<cancel>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<transfer>

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>

Group element containing job information.

<jobName>

Specifies logical job identifier.

File transfer cancel message format

A file transfer request returns a 48-character ID that identifies the transfer for a specific agent. This ID is used to cancel transfers.

Understanding the transfer cancel message

The elements and attributes used in transfer cancel messages are described:

<cancel>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<transfer>

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>

Optional. Group element containing job information.

<jobName>

Specifies logical job identifier.

Examples

Examples of XML messages that conform to this schema are provided for each of the following requests:

- [Create a file transfer](#)
- [Create an asynchronous file transfer request](#)
- [Cancel a file transfer](#)
- [Create a scheduled transfer](#)
- [Delete a scheduled transfer](#)
- [Create a managed call](#)
- [Create a file transfer that includes managed calls](#)

File transfer request message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

Create transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
version="4.00"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <destinationAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <transferSet>
      <item checksumMethod="MD5" mode="binary">
        <source disposition="leave" recursive="false">
          <file>/etc/passwd</file>
        </source>
        <destination exist="overwrite" type="directory">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Create transfer request - synchronous

When a user requests a blocking synchronous request, that is, they wait for the transfer to complete and receive status messages, the message placed on the command queue contains a reply element that specifies the queue that a reply message is sent to. The following example shows the message placed on the command queue used by FTEAGENT:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
```

```

<managedTransfer>
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <sourceAgent agent="FTEAGENT"
    QMgr="QM1"/>
  <destinationAgent agent="AGENT2"
    QMgr="QM2"/>
  <reply QMGR="QM1">WMQFTE.492D0D5502770020</reply>
  <transferSet>
    <item mode="binary" checksumMethod="MD5">
      <source recursive="false" disposition="leave">
        <file>c:\sourcefiles\source1.doc</file>
      </source>
      <destination type="file" exist="overwrite">
        <file>c:\destinationfiles\dest1.doc</file>
      </destination>
    </item>
  </transferSet>
</managedTransfer>
</request>

```

The <reply> element is populated with the name of the command queue manager where a temporary dynamic queue has been created to receive reply about the successful (or otherwise) completion of the transfer. The name of the temporary dynamic queue is composed of two parts:

- The prefix as defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file (it is WMQFTE. by default)
- The ID of the queue as generated by IBM MQ

Cancel transfer request

```

<?xml version="1.0" encoding="UTF-8"?>
<cancel xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <transfer>414D51205553322E42494E444494E47538B0F404D032C0020</transfer>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20002007</reply>
</cancel>

```

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

Scheduled file transfer message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

Create scheduled transfer

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>

```

```

<schedule>
  <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00</submit>
</schedule>
<sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
<destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
<transferSet>
  <item checksumMethod="MD5" mode="binary">
    <source disposition="leave" recursive="false">
      <file>/etc/passwd</file>
    </source>
    <destination exist="overwrite" type="directory">
      <file>/tmp</file>
    </destination>
  </item>
</transferSet>
</managedTransfer>
</request>

```

Delete scheduled transfer

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <deleteScheduledTransfer>
    <originator>
      <delete>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
      </delete>
    </originator>
    <ID>1</ID>
    <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003902</reply>
  </deleteScheduledTransfer>
</request>

```

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

MFT agent call request message examples

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

Managed call request example

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <agent agent="DNWE" QMgr="QM1"/>
    <transferSet>
      <call>
        <command name="echo" successRC="0">
          <argument>call</argument>
          <argument>test</argument>
        </command>
      </call>
    </transferSet>
  </job>

```

```

        <name>managedCallCalls.xml</name>
    </job>
</managedCall>
</request>

```

Managed transfer request example with calls

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent agent="DNWE" QMgr="QM1" />
    <destinationAgent agent="DNWE" QMgr="QM1" />
    <transferSet>
      <preSourceCall>
        <command name="echo" successRC="0">
          <argument>preSourceCall</argument>
          <argument>test</argument>
        </command>
      </preSourceCall>
      <postSourceCall>
        <command name="echo" successRC="0">
          <argument>postSourceCall</argument>
          <argument>test</argument>
        </command>
      </postSourceCall>
      <preDestinationCall>
        <command name="echo" successRC="0">
          <argument>preDestinationCall</argument>
          <argument>test</argument>
        </command>
      </preDestinationCall>
      <postDestinationCall>
        <command name="echo" successRC="0">
          <argument>postDestinationCall</argument>
          <argument>test</argument>
        </command>
      </postDestinationCall>
    </transferSet>
  <job>
    <name>managedTransferCalls.xml</name>
  </job>
</managedTransfer>
</request>

```

Related concepts

[“Specifying programs to run with MFT” on page 239](#)

You can run programs on a system where a Managed File Transfer Agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

MFT monitor request message formats

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the `fteCreateMonitor` command or using the IBM MQ Explorer interface.

The monitor XML must conform to the `Monitor.xsd` schema using the `<monitor>` element as the root element.

Monitor messages can have one of the following root elements:

- `<monitor>` - for creating and starting a new resource monitor
- `<deleteMonitor>` - for stopping and deleting an existing monitor

There is no command message for the `fteListMonitors` command because the command directly retrieves matching monitor definitions from the `SYSTEM.FTE` topic.

Schema

The following schema describes which elements are valid in a monitor request XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
            targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/
MonitorDefinition"
            xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

<xsd:include schemaLocation="FileTransfer.xsd"/>

  <xsd:element name="monitor">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="monitorNameType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string"
                    minOccurs="0" maxOccurs="1"/>
        <xsd:element name="pollInterval" type="pollIntervalType"
                    minOccurs="1" maxOccurs="1"
                    default="10"/>
        <xsd:element name="batch" type="batchType"
                    minOccurs="0" maxOccurs="1"/>
        <xsd:element name="agent" type="agentNameType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="resources" type="monitorResourcesType"
                    minOccurs="0" maxOccurs="1"/>
        <xsd:element name="triggerMatch" type="triggerMatchType"
                    maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType"
                    maxOccurs="1" minOccurs="0"/>
        <xsd:element name="tasks" type="monitorTasksType"
                    maxOccurs="1" minOccurs="1"/>
        <xsd:element name="originator" type="origRequestType"
                    maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType"
                    maxOccurs="1" minOccurs="0"/>
        <xsd:element name="defaultVariables" type="defaultVariablesType"
                    maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="deleteMonitor">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="monitorNameType"
                    minOccurs="1" maxOccurs="1"/>
        <xsd:element name="originator" type="origRequestType"
                    maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType"
                    maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

<xsd:complexType name="transferRequestType">
  <xsd:choice>
    <xsd:element name="managedTransfer" type="managedTransferType"/>
    <xsd:element name="managedCall" type="managedCallType"/>
  </xsd:choice>
  <xsd:attribute name="version" type="versionType"/>
</xsd:complexType>

<xsd:complexType name="monitorResourcesType">
  <xsd:choice>
    <xsd:sequence>
      <xsd:element name="directory" type="monitoredDirectoryType"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:element name="queue" type="monitoredQueueType"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="monitoredDirectoryType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="recursionLevel"
type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="id" type="resourceIdAttrType"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="monitoredQueueType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="id" type="resourceIdAttrType"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="triggerMatchType">
  <xsd:sequence>
    <xsd:element name="conditions" type="conditionsType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="conditionsType">
  <xsd:choice minOccurs="1">
    <xsd:element name="allOf" type="listPredicateType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="anyOf" type="listPredicateType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="condition" type="conditionType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="listPredicateType">
  <xsd:choice>
    <xsd:element name="condition" type="conditionType"
      minOccurs="1" maxOccurs="unbounded"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="conditionType">
  <xsd:sequence>
    <xsd:element name="name" type="conditionNameType"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="resource" type="resourceIdType"
      minOccurs="0" maxOccurs="1"/>
    <xsd:choice minOccurs="1">
      <xsd:element name="fileMatch"
type="fileMatchConditionType"
      minOccurs="1" maxOccurs="1"/>
      <xsd:element name="fileNoMatch"
type="fileNoMatchConditionType"
      minOccurs="1"
maxOccurs="1"/>
      <xsd:element name="fileSize"
type="fileSizeConditionType"
      minOccurs="1" maxOccurs="1"/>
      <xsd:element name="queueNotEmpty"
type="queueNotEmptyConditionType"
      minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

```

```

        <xsd:element name="completeGroups"
type="completeGroupsConditionType"
        minOccurs="1"          maxOccurs="1"/>
        <xsd:element name="fileSizeSame"      type="fileSizeSameType"
        minOccurs="1"          maxOccurs="1"/>
    </xsd:choice>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileNoMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileSizeConditionType">
    <xsd:sequence>
        <xsd:element name="compare" type="sizeCompareType"
minOccurs="1" default="0" />
        <xsd:element name="pattern" type="conditionPatternType"
minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="sizeCompareType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="operator" type="sizeOperatorType"
use="required" />
            <xsd:attribute name="units" type="fileSizeUnitsType"
use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="sizeOperatorType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value=">=" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="fileSizeUnitsType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[bB] | [kK] [bB] | [mM] [bB] | [gG] [bB]" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionPatternType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="type" type="patternTypeAttributeType"
use="optional" default="wildcard" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="patternTypeAttributeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="regex" />
        <xsd:enumeration value="wildcard" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionNameType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string" />
    </xsd:simpleContent>
</xsd:complexType>

```

```

<xsd:complexType name="queueNotEmptyConditionType"/>
<xsd:complexType name="completeGroupsConditionType"/>
<xsd:complexType name="fileSizeSameType">
  <xsd:sequence>
    <xsd:element name="pattern" type="conditionPatternType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="exclude" type="conditionPatternType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="polls" type="positiveIntegerType" use="required"/>
</xsd:complexType>
<xsd:complexType name="pollIntervalType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:int">
      <xsd:attribute name="units" type="timeUnitsType"
        use="optional" default="minutes"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="batchType">
  <xsd:attribute name="maxSize" type="positiveIntegerType" use="required"/>
</xsd:complexType>
<xsd:simpleType name="timeUnitsType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="seconds"/>
    <xsd:enumeration value="minutes"/>
    <xsd:enumeration value="hours"/>
    <xsd:enumeration value="days"/>
    <xsd:enumeration value="weeks"/>
    <xsd:enumeration value="months"/>
    <xsd:enumeration value="years"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="monitorTasksType">
  <xsd:sequence>
    <xsd:element name="task" type="monitorTaskType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="monitorTaskType">
  <xsd:sequence>
    <xsd:element name="name" type="monitorTaskNameType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="description" type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="transfer" type="transferTaskType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="transferTaskType">
  <xsd:sequence>
    <xsd:element name="request" type="transferRequestType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="resourceIdType">
  <xsd:attribute name="id" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:simpleType name="resourceIdAttrType">
  <xsd:restriction base="xsd:string"></xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="monitorNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="^[%]*"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="agentNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[%_0-9A-Z]*"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="monitorTaskNameType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value=".*"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="defaultVariablesType">
    <xsd:sequence>
      <xsd:element name="variable" type="variableType"
        maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="variableType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="key" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>

```

Understanding the create monitor message

The elements and attributes used in create monitor messages are described:

Element descriptions

<monitor>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<name>

The name of the monitor, unique within the monitor's agent.

<description>

Description of the monitor (not currently used).

<pollInterval>

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: <ul style="list-style-type: none"> • seconds • minutes • hours • days • weeks • months • years

<agent>

Name of the agent the monitor is associated with.

<resources>

Group element that contains the elements specifying the resources to monitor.

<directory>

Fully qualified path specifying the directory on the monitor's agent machine to monitor.

Attribute	Description
recursionLevel	The number of subdirectories to monitor in addition to the specified directory.
id	Unique identifier for the resource.

<queue>

Queue name specifying the queue to monitor on the monitoring agent's queue manager.

<triggerMatch>

Group element that contains the elements specifying the trigger conditions to compare with the monitored resource.

<conditions>

Group element that contains the elements specifying the type of condition to compare with the monitored resource.

<allOf>

Predicate that specifies that all contained conditions must be satisfied.

<anyOf>

Predicate that specifies that any contained conditions must be satisfied.

<condition>

Defines a comparison condition that will contribute to the overall monitor trigger condition.

<name>

Name of the condition.

<resource>

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:

- queueNotEmpty
- completeGroups

<fileMatch>

Group element for a file name match condition.

<pattern>

Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

<fileNoMatch>

Group element for an inverse file name match condition.

<pattern>

Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

<fileSize>

Group element for a file size comparison.

<compare>

Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only '>=' is supported.
units	<p>Specifies file size units, which can be one of:</p> <ul style="list-style-type: none"> • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes <p>The units value is case insensitive, so 'mb' works as well as 'MB'.</p>

<pattern>

File name pattern to match. Default is '*' (any file will match).

<queueNotEmpty>

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

<completeGroups>

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

<reply>

Optional element that is used to specify reply queue for asynchronous requests.

Attribute	Description
QMGR	Queue manager name.

<tasks>

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

<task>

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

<name>

Name of the task. Accepts any alphanumeric characters.

<description>

Description of the task. Any text value is allowed.

<transfer>

Group element that defines a transfer task.

<request>

Group element that defines the type of task. This must contain one of the following elements which are inherited from the `FileTransfer.xsd` schema definition:

- [managedTransfer](#)
- `managedCall`

Attribute	Description
version	Version of the request as provided by Managed File Transfer. This is in the form n.mm where n is the major release version and mm is the minor version. For example 1.00.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<job>

Group element containing job information.

<jobName>

Specifies logical job identifier.

<defaultVariables>

Group element containing one or more variable elements. These variables are used in variable substitution when monitoring a queue. For more information about variable substitution, see [“Customizing MFT resource monitor tasks with variable substitution” on page 172.](#)

<variable>

Element containing the value associated with the key given by the key attribute.

Attribute	Description
key	The name of the default variable.

Understanding the delete monitor message

The elements and attributes used in delete monitor messages are described:

Element descriptions

<deleteMonitor>

Group element containing all the elements required to stop and delete a monitor.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<name>

Name of monitor to delete.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<reply>

Specifies the name of the temporary reply queue generated for the request. The name of the queue is as defined by the key `dynamicQueuePrefix` in the `command.properties` configuration file. If this is not specified, the queue name has a default value of `WMQFTE`.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor requests:

- [Create a monitor](#)
- [Delete a monitor](#)

MFT monitor request message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

Create monitor request

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
  <pollInterval>1</pollInterval>
  <agent>US2.BINDINGS.FILE</agent>
  <resources>
    <directory recursionLevel="0">/srv/nfs/incoming</directory>
  </resources>
  <triggerMatch>
    <conditions>
      <allOf>
        <condition>
          <fileMatch>
            <pattern>*.completed</pattern>
          </fileMatch>
        </condition>
      </allOf>
    </conditions>
  </triggerMatch>
  <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003702</reply>
  <tasks>
    <task>
      <name/>
      <transfer>
        <request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          version="4.00"
          xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
          <managedTransfer>
            <originator>
              <hostName>example.com.</hostName>
              <userID>mqm</userID>
            </originator>
            <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
            <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
            <transferSet>
              <item checksumMethod="MD5" mode="binary">
                <source disposition="leave" recursive="false">
                  <file>/srv/nfs/incoming/*.txt</file>
                </source>
                <destination exist="error" type="directory">
                  <file>/srv/backup</file>
                </destination>
              </item>
            </transferSet>
          </managedTransfer>
        </request>
      </transfer>
    </task>
  </tasks>
  <originator>
    <hostName>example.com.</hostName>
  </originator>
</monitor>
```

```

    <userID>mqm</userID>
  </originator>
</monitor:monitor>

```

Delete monitor request

```

<?xml version="1.0" encoding="UTF-8"?>
<monitor:deleteMonitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003705</reply>
</monitor:deleteMonitor>

```

Related reference

[“MFT monitor request message formats” on page 772](#)

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the `fteCreateMonitor` command or using the IBM MQ Explorer interface.

Ping MFT agent request message format

You can ping an agent by issuing an `ftePingAgent` command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the `PingAgent.xsd` schema. After you have installed Managed File Transfer, you can find the `PingAgent.xsd` schema file in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/schema`. The `PingAgent.xsd` schema imports `fteutils.xsd`, which is in the same directory.

When the agent receives a ping agent request message on its command queue, if the agent is active, it returns an XML response message to the command or application that put the ping agent request message on the command queue. The response message from the agent is in the format defined by `Reply.xsd`. For more information about this format, see [“MFT agent reply message format” on page 782](#).

Schema

The following schema describes which elements are valid in an ping agent request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="pingAgent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Understanding the ping agent request message

The elements and attributes used in the ping agent request messages are described in the following list:

<pingAgent>

Group element containing all the elements required to specify a ping agent request.

<originator>

Group element containing all the elements required to specify the originator of the ping request.

<hostName>

The host name of the machine where the request originated.

<userID>

The user name of the originator of the request.

<mqmdUserID>

The MQMD user name of the originator of the request.

<agent>

The agent to ping.

Attribute	Description
agent	Required. The name of the agent.
QMGr	Optional. The queue manager that the agent connects to.

<reply>

The name of the queue for the agent to send the reply message to.

Attribute	Description
QMGR	Required. The name of the queue manager where the reply queue is located.

Example

This example shows a ping agent message sent to the agent AGENT_JUPITER. If AGENT_JUPITER is active and able to process agent requests, it sends a response message to the queue WMQFTE.4D400F8B20003708 on QM_JUPITER.

```
<?xml version="1.0" encoding="UTF-8"?>
<ping:pingAgent xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:ping="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  version="4.00">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <agent agent="AGENT_JUPITER" QMGr="QM_JUPITER"/>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003708</reply>
</ping:pingAgent>
```

MFT agent reply message format

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the Reply.xsd schema. The Reply.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The Reply.xsd schema imports `fteutils.xsd`, which is in the same directory.

Schema

The following schema describes which elements are valid in a reply XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="TransferLog.xsd"/>
  <xsd:element name="reply">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="transferSet" type="transferSetType" minOccurs="0"/>
```


Every time a user submits a request to perform a restricted action to the agent, either by using an Managed File Transfer command or by using the IBM MQ Explorer plugin, the agent checks that the user has the authority to perform the action. If the user fails that authority check, a not authorized log message is published to the coordination queue manager on its SYSTEM.FTE/Log/agent_name/NotAuthorized topic.

This message conforms to the TransferLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<notAuthorized version="3.00"
  ID="414d5120716d312020202020202020204da5924a2010ce03"
  agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2009-08-28T12:31:15.781Z">not_authorized</action>
  <originator>
    <mqmdUserID>test1</mqmdUserID>
  </originator>
  <authority>administration</authority>
  <status resultCode="53">
    <supplement>BFGCH0083E: The user (test1) does not have the authority (ADMINISTRATION) required
to shut down agent 'AGENT'.</supplement>
    <supplement>
      &lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
&lt;internal:request version=&quot;3.00&quot; xmlns:xsi=&quot;https://www.w3.org/2001/XMLSchema-
instance&quot;
  xmlns:internal=&quot;http://wmqfte.ibm.com/internal&quot;&gt;
&lt;internal:shutdown agent=&quot;SYSTEM.FTE.COMMAND.AGENT&quot; hostname= &quot;qm1&quot;
mode=&quot;controlled&quot;/&gt;
&lt;reply QMGR=&quot;qm1&quot;&gt;WMQFTE.4A92A54D02CE1020&lt;/reply&gt;
&lt;/internal:request&gt;
    </supplement>
  </status>
</notAuthorized>
```

This message is a log of the following information:

- Who originated the request
- The level of Managed File Transfer access authority required to perform the request
- The status of the request
- The request specification

Understanding the not authorized log message

The elements and attributes used in the not authorized message are described:

<notAuthorized>

Group element that describes a single failed user authorization check.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.
ID	The unique identifier for the request that was not authorized.

<originator>

Group element that contains the elements specifying the originator of the request.

<authority>

Specifies the level of Managed File Transfer access authority that the user required to perform the requested action.

<mqmdUserID>

The IBM MQ user ID that was supplied in the message descriptor (MQMD)

<action>

Specifies the authorization status of the request matching the ID attribute of <notAuthorized> element.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

<status>

The result code and supplement messages.

MFT credentials file format

The MQMFTCredentials.xml file contains sensitive user ID and password information. The elements in the MQMFTCredentials.xml file must conform to the MQMFTCredentials.xsd schema. The security of credentials files is the responsibility of the user.

V 9.0.0.2 **V 9.0.3** From IBM MQ 9.0.0 Fix Pack 2 and IBM MQ 9.0.3, you can disable the default compatibility mode and enable MQCSP authentication for a Managed File Transfer agent by adding a new parameter, **useMQCSPAuthentication**, to the MFT credentials file MQMFTCredentials.xml for the relevant user and setting it to true. For more information, see [Enabling MQCSP authentication mode](#).

z/OS **z/OS** From IBM MQ 8.0, the MQMFTCredentials.xsd file can also be a PDSE member on z/OS.

Before IBM WebSphere MQ 7.5 the information contained in the MQMFTCredentials.xml file was held in separate properties files.

The MQMFTCredentials.xml file must conform to the MQMFTCredentials.xsd schema. The MQMFTCredentials.xml schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the Managed File Transfer installation.

Schema

The following schema describes which elements are valid in the MQMFTCredentials.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  @start_non_restricted_prolog@
  Version: %Z% %I% %W% %E% %U% [%H% %T%]

  Licensed Materials - Property of IBM

  5724-H72

  Copyright IBM Corp. 2012, 2024. All Rights Reserved.

  US Government Users Restricted Rights - Use, duplication or
  disclosure restricted by GSA ADP Schedule Contract with
  IBM Corp.
  @end_non_restricted_prolog@
-->

<!--
  This schema defines the format of an MQMFTCredentials file. Files of this type
  store credential information for agent and logger processes. They can contain
  user names and passwords either in clear text or which have been obfuscated
  using the fteObfuscate command.
-->

<!-- Example mqmftCredentials.xml file:
<?xml version="1.0" encoding="UTF-8"?>
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/
MQMFTCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/MQMFTCredentials MQMFTCredentials.xsd">

  <tns:logger name="LOG1" user="user1" password="passw0rd"/>
  <tns:logger name="ORACLE" userCipher="kj2h3dfkgf" passwordCipher="1a3n67eae"/>
  <tns:file path="/home/emma/trust.jks" password="passw0rd"/>
  <tns:file path="/var/tmp/keystore.jks" passwordCipher="e71vKCg2pf"/>

  <tns:qmgr name="QM_COORD" user="tim" mqUserId="user1" mqPassword="passw0rd"/>
  <tns:qmgr name="QM_COORD" user="tom" mqUserId="user1" mqPasswordCipher="e71vKCg2pf"/>
```

```

V 9.0.3 <tns:qmgr name="QM_COORD" user="ernest" mqUserId="ernest"
mqPassword="AveryL0ngPassw0rd2135" useMQCSPAAuthentication="true"/>
  </tns:mqmftCredentials>
-->

<schema targetNamespace="http://wmqfte.ibm.com/MQMFTCredentials"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials">

  <element name="mqmftCredentials" type="tns:mqmftCredentialsType"/>

  <complexType name="mqmftCredentialsType">
    <sequence>
      <choice minOccurs="0" maxOccurs="unbounded">
        <element name="logger" type="tns:loggerType"/>
        <element name="file" type="tns:fileType"/>
        <element name="qmgr" type="tns:mqUserPassType"/>
      </choice>
    </sequence>
  </complexType>

  <complexType name="loggerType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="user" type="string" use="optional"/>
    <attribute name="userCipher" type="string" use="optional"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="fileType">
    <attribute name="path" type="string" use="required"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <!-- Example XML:

  <tns:qmgr name="QM_COORD" user="tim" mqUserId="user1" mqPassword="passw0rd"/>
  <tns:qmgr name="QM_COORD" user="tom" mqUserIdCipher="xh5U7812x"
mqPasswordCipher="e71vKCg2pf"/>
  <tns:qmgr name="QM_COORD" mqUserId="defaultUser" mqPassword="passw0rd"/>
V 9.0.3 <tns:qmgr name="QM_COORD" user="ernest" mqUserId="ernest"
mqPassword="AveryL0ngPassw0rd2135" useMQCSPAAuthentication="true"/>
-->

  <complexType name="mqUserPassType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="user" type="string" use="optional"/>
    <attribute name="mqUserId" type="string" use="optional"/>
    <attribute name="mqUserIdCipher" type="string" use="optional"/>
    <attribute name="mqPassword" type="string" use="optional"/>
    <attribute name="mqPasswordCipher" type="string" use="optional"/>
V 9.0.3 <attribute name="useMQCSPAAuthentication" type="boolean" use="optional"/>
  </complexType>

</schema>

```

Understanding the MQMFTCredentials.xml file

The elements and attributes used in the MQMFTCredentials.xml file are described in the following list.

<mqmftCredentials>

The root element of the XML document.

<file>

The file in the transfer.

Attribute	Description
path	Path to the key or truststore file being accessed.
password	Password to access the file.

<logger>

The logger responsible for logging activity.

Attribute	Description
name	The name of the logger.
user	The user name the logger will use to connect to its database.
password	The password the logger will use to connect to its database.

<qmgr>

The IBM MQ queue manager connection.

Attribute	Description
name	The name of the associated IBM MQ queue manager.
user	Optional: The name of user requesting the connection.
mqUserId or mqUserIdCipher	The clear text user ID (mqUserId), or obfuscated text user ID (mqUserIdCipher) to supply to an IBM MQ queue manager.
mqPassword or mqPasswordCipher	The clear text password (mqPassword), or obfuscated text password (mqPasswordCipher) to supply to an IBM MQ queue manager.

Note: The `MQMFTCredentials.xml` file can contain sensitive information, so when it is created ensure that the file permissions are reviewed. When using a sandbox, set it to be excluded. For more information on sandboxes, see [“Working with MFT agent sandboxes”](#) on page 45.

Related concepts

[“Configuring MQMFTCredentials.xml on z/OS”](#) on page 42

If Managed File Transfer (MFT) is configured with security enabled, connection authentication requires all MFT agents, and commands that connect to a queue manager, to supply user ID and password credentials.

Related reference

[“MFT and IBM MQ connection authentication”](#) on page 39

Connection authentication allows a queue manager to be configured to authenticate applications by using a provided user ID and password. If the associated queue manager has security enabled, and requires credential details (user ID and password), the connection authentication feature must be enabled before a successful connection to a queue manager can be made. Connection authentication can be run in compatibility mode or MQCSP authentication mode.

[“fteObfuscate: encrypt sensitive data”](#) on page 473

The **`fteObfuscate`** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file.

Protocol bridge credentials file format

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

The `ProtocolBridgeCredentials.xml` file must conform to the `ProtocolBridgeCredentials.xsd` schema. The `ProtocolBridgeCredentials.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MQMFT installation. Users are responsible for manually creating the `ProtocolBridgeCredentials.xml` file, it is no longer created by the **`fteCreateBridgeAgent`** command. Sample files are available in the `MQ_INSTALLATION_PATH/mqft/samples` directory of the MQMFT installation.

IBM WebSphere MQ 7.5 introduced a new `<agent>` element that contains the `<server>` or `<serverHost>` element for the named agent.

The ProtocolBridgeCredentials.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema - 7.5 or later

The following schema describes which elements are valid in the ProtocolBridgeCredentials.xml file for IBM MQ 8.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeCredentials" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/
ProtocolBridgeCredentials">

  <!--
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
  ProtocolBridgeCredentials.xsd">
    <tns:agent name="agent1">
      <tns:serverHost name="myserver">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:serverHost>
    </tns:agent>

    <tns:agent name="agent2">
      <tns:server name="server*" pattern="wildcard">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:server>
    </tns:agent>

    <tns:agent name="agent3">
      <tns:serverHost name="ftpsServer"
        keyStorePassword="keypass"
        trustStorePassword="trustpass">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
      </tns:serverHost>
    </tns:agent>

  </tns:credentials>
  -->

  <element name="credentials" type="tns:credentialsType"/>

  <complexType name="credentialsType">
    <sequence>
      <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="agentType">
    <choice minOccurs="0" maxOccurs="1">
      <element name="serverHost" type="tns:serverHostType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="server" type="tns:serverType" minOccurs="0" maxOccurs="unbounded"/>
    </choice>
    <attribute name="name" type="string" use="required"/>
  </complexType>

  <complexType name="serverHostType">
    <sequence>
      <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="keyStorePassword" type="string" use="optional"/>
    <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
    <attribute name="trustStorePassword" type="string" use="optional"/>
  </complexType>

```

```

    <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
</complexType>

<complexType name="serverType">
  <sequence>
    <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="keyStorePassword" type="string" use="optional"/>
  <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
  <attribute name="trustStorePassword" type="string" use="optional"/>
  <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
</complexType>

<element name="user" type="tns:userType"/>

<complexType name="userType">
  <sequence>
    <element ref="tns:privateKey" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="serverUserId" type="string" use="optional"/>
  <attribute name="serverUserIdCipher" type="string" use="optional"/>
  <attribute name="serverPassword" type="string" use="optional"/>
  <attribute name="serverPasswordCipher" type="string" use="optional"/>
  <attribute name="hostKey" use="optional">
    <simpleType>
      <restriction base="string">
        <pattern
          value="([a-zA-Z0-9]){2}(:([a-zA-Z0-9]){2})*">
        </pattern>
      </restriction>
    </simpleType>
  </attribute>
</complexType>

<element name="privateKey" type="tns:privateKeyType"/>

<complexType name="privateKeyType">
  <simpleContent>
    <extension base="string">
      <attribute name="keyPassword" type="string" use="optional"/>
      <attribute name="keyPasswordCipher" type="string" use="optional"/>
      <attribute name="associationName" type="string" use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
  Determines the type of pattern matching to use.
-->
<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>
</schema>

```

Understanding the ProtocolBridgeCredentials.xml file

The elements and attributes used in the ProtocolBridgeCredentials.xml file are described in the following list.

<credentials>

Group element containing elements that describe the credentials used by a protocol bridge agent to connect to a protocol server.

<agent>

Element containing a <server> or <serverHost> definition for a named agent.

<server>

The protocol server that the protocol bridge connects to.

The <server> element is not supported for 7.0.4 or earlier.

Attribute	Description
name	The name of the protocol server.
pattern	If you have used wildcards or regular expressions to specify the pattern of a protocol server name, use either <code>wildcard</code> or <code>regex</code> .
trustStorePassword or trustStorePasswordCipher	Required when the <code><server></code> element refers to an FTPS server. The password used to access the truststore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<serverHost>

The host name of the protocol server that the protocol bridge connects to.

The `ProtocolBridgeCredentials.xml` file can either contain `<serverHost>` elements or `<server>` elements but you cannot use a mixture of the two different types. When you use `<serverHost>`, the name is matched against the protocol server's host name. When you use `<server>`, the name is matched against the protocol server's name (as defined in the `ProtocolBridgeProperties.xml` file).

Attribute	Description
name	The host name or IP address of the protocol server.
trustStorePassword or trustStorePasswordCipher	Required when the <code><serverHost></code> element refers to an FTPS server. The password used to access the truststore. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. This property is optional unless you set the <code>keyStore</code> attribute, in which case it is required. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<user>

A user mapping from a Managed File Transfer user name to a protocol server user name.

Attribute	Description
name	The user name that is used with Managed File Transfer.
serverUserId or serverUserIdCipher	The user name that is used with the protocol server. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
serverPassword or serverPasswordCipher	The password for the user name used on the protocol server. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
hostKey	The server host SSH fingerprint.

<privateKey>

The private key of a user.

Attribute	Description
keyPassword or keyStorePasswordCipher	The password for the private key. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
associationName	A name used for trace and logging.

Protocol bridge properties file format

The ProtocolBridgeProperties.xml file in the agent configuration directory defines properties for protocol file servers.

The ProtocolBridgeProperties.xml file must conform to the ProtocolBridgeProperties.xsd schema. The ProtocolBridgeProperties.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the Managed File Transfer installation. A template file, ProtocolBridgeProperties.xml, is created by the **fteCreateBridgeAgent** command in the agent configuration directory.

The ProtocolBridgeProperties.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema

The following schema describes the ProtocolBridgeProperties.xml file.

Note: The `maxReconnectRetry` and `reconnectWaitPeriod` attributes are not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2, or later.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeProperties" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties">
  <!--
    Example: ProtocolBridgeProperties.xml
  -->
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
      ProtocolBridgeProperties.xsd">
    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml"/>
    <tns:defaultServer name="myserver"/>
    <tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234" platform="windows"
      timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
      listFormat="unix" limitedWrite="false"/>
    <tns:sftpServer name="server1" host="myhost.hursley.ibm.com" platform="windows"
      fileEncoding="UTF-8" limitedWrite="false">
      <limits maxListFileNames="10"/>
    </tns:sftpServer>
  </tns:serverProperties>
  -->

  <!-- Root element for the document -->
  <element name="serverProperties" type="tns:serverPropertiesType"></element>

  <!--
    A container for all protocol bridge server properties
  -->
  <complexType name="serverPropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1"/>
      <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
        <element name="ftpServer" type="tns:ftpServerType"/>
        <element name="sftpServer" type="tns:sftpServerType"/>
        <element name="ftpsServer" type="tns:ftpsServerType"/>
        <element name="ftpsfgServer" type="tns:ftpsfgServerType"/>
        <element name="ftpsfgServer" type="tns:ftpsfgServerType"/>
      </choice>
    </sequence>
  </complexType>
```

```

<!--
  A container for a server name
-->
<complexType name="serverName">
  <attribute name="name" type="tns:serverNameType" use="required"/>
</complexType>

<!--
  A container for a credentials file name
-->
<complexType name="credentialsFileName">
  <attribute name="path" type="string" use="required"/>
</complexType>

<!--
  A container for all the information about an FTP server
-->
<complexType name="ftpServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpServerAttributes"/>
  <attribute name="passiveMode" type="boolean" use="optional"/>
</complexType>

<!--
  A container for all the information about an SFG FTP server
-->
<complexType name="ftpsfgServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpServerAttributes"/>
</complexType>

<!--
  A container for all the information about an SFTP server
-->
<complexType name="sftpServerType">
  <sequence>
    <element name="limits" type="tns:sftpLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:sftpServerAttributes"/>
</complexType>

<!--
  A container for all the information about a FTPS server
-->
<complexType name="ftpsServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  A container for all the information about a SFG FTPS server
-->
<complexType name="ftpsfgServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  Attributes common to all server types
-->
<attributeGroup name="generalServerAttributes">
  <attribute name="name" type="tns:serverNameType" use="required"/>
  <attribute name="host" type="string" use="required"/>
  <attribute name="port" type="nonNegativeInteger" use="optional"/>
  <attribute name="platform" type="tns:platformType" use="required"/>
  <attribute name="fileEncoding" type="string" use="required"/>
  <attribute name="limitedWrite" type="boolean" use="optional"/>
  <attribute name="controlEncoding" type="string" use="optional"/>
</attributeGroup>

<!--
  Attributes common to ftp and ftps server types
-->

```

```

<attributeGroup name="ftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes"/>
  <attribute name="timeZone" type="string" use="required"/>
  <attribute name="locale" type="tns:localeType" use="required"/>
  <attribute name="listFormat" type="tns:listFormatType" use="optional"/>
  <attribute name="listFileRecentDateFormat" type="tns:dateFormatType" use="optional"/>
  <attribute name="listFileOldDateFormat" type="tns:dateFormatType" use="optional"/>
  <attribute name="monthShortNames" type="tns:monthShortNamesType" use="optional"/>
</attributeGroup>

<!--
  Attributes common to ftps server types
-->
<attributeGroup name="ftpsServerAttributes">
  <attributeGroup ref="tns:ftpServerAttributes"/>
  <attribute name="ftpsType" type="tns:ftpsTypeType" use="optional"/>
  <attribute name="trustStore" type="string" use="required"/>
  <attribute name="trustStoreType" type="string" use="optional"/>
  <attribute name="keyStore" type="string" use="optional"/>
  <attribute name="keyStoreType" type="string" use="optional"/>
  <attribute name="ccc" type="boolean" use="optional"/>
  <attribute name="protFirst" type="boolean" use="optional"/>
  <attribute name="auth" type="string" use="optional"/>
  <attribute name="connectTimeout" type="nonNegativeInteger" use="optional"/>
</attributeGroup>

<!--
  A container for limit-type attributes for a server. Limit parameters
  are optional, and if not specified a system default will be used.
-->
<complexType name="generalLimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
</complexType>

<complexType name="sftpLimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
  <attribute name="connectionTimeout" type="nonNegativeInteger" use="optional"/>
</complexType>

<!--
  Attributes for limits common to all server types
-->
<attributeGroup name="generalLimitAttributes">
  <attribute name="maxListFileNames" type="positiveInteger" use="optional"/>
  <attribute name="maxListDirectoryLevels" type="nonNegativeInteger" use="optional"/>
  <attribute name="maxReconnectRetry" type="nonNegativeInteger" use="optional"/>
  <attribute name="reconnectWaitPeriod" type="nonNegativeInteger" use="optional"/>
  <attribute name="maxSessions" type="positiveInteger" use="optional"/>
  <attribute name="socketTimeout" type="nonNegativeInteger" use="optional"/>
</attributeGroup>

<!--
  The type for matching valid server names. Server names must be at least 2 characters in length
and
  are limited to alphanumeric characters and the following characters: ".", "_", "/" and "%".
-->
<simpleType name="serverNameType">
  <restriction base="string">
    <pattern value="[0-9a-zA-Z\._/%]{2,}" />
  </restriction>
</simpleType>

<!--
  The types of platform supported.
-->
<simpleType name="platformType">
  <restriction base="string">
    </restriction>
</simpleType>

<!--
  The type for matching a locale specification.
-->
<simpleType name="localeType">
  <restriction base="string">
    <pattern value="(.)[-_](.)/" />
  </restriction>
</simpleType>

<!--
  The types of list format supported (for FTP servers).

```

```

-->
<simpleType name="listFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
  Date format for FTP client directory listing on an FTP server. This is
  the format to be passed to methods setDefaultDateFormatStr and
  setRecentDateFormatStr for Java class:
  org.apache.commons.net.ftp.FTPClientConfig
-->
<simpleType name="dateFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
  A list of language-defined short month names can be specified. These are
  used for translating the directory listing received from the FTP server.
  The format is a string of three character month names separated by "|"
-->
<simpleType name="monthShortNamesType">
  <restriction base="string">
    <pattern value="(...\|){11}{...}" />
  </restriction>
</simpleType>

<!--
  The enumerations of the allowed FTPS types: "implicit" & "explicit"
  If not specified the default is "explicit"
-->
<simpleType name="ftpsTypeType">
  <restriction base="string">
    <enumeration value="explicit" />
    <enumeration value="implicit" />
  </restriction>
</simpleType>

<!--
  Attribute Group for SFTP Servers
-->
<attributeGroup name="sftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes" />
  <attribute name="cipherList" type="string" use="optional" />
</attributeGroup>
</schema>

```

Understanding the ProtocolBridgeProperties.xml file

The elements and attributes that are used in the ProtocolBridgeProperties.xml file are described in the following list:

<serverProperties>

Root element of the XML document

<credentialsFile>

Path to the file containing credentials. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see [“The use of environment variables in MFT properties” on page 511](#)

<defaultServer>

The protocol file server that acts as the default server for file transfers

<ftpServer>

An FTP file server

<sftpServer>

An SFTP file server

<ftpsServer>

An FTPS file server

General server attributes that apply to all types of protocol file server:

Attribute	Description
name	Required. The name of the protocol file server. Protocol server names must be at least two characters in length, are not case-sensitive, and are limited to alphanumeric characters and the following characters: <ul style="list-style-type: none"> • period (.) • underscore (_) • forward slash (/) • percent sign (%)
host	Required. The host name or IP address of the protocol file server that you want to send files to or receive files from.
port	Optional. The port number of the protocol file server that you want to send files to or receive files from.
platform	Required. The platform of the protocol file server that you want to send files to or receive files from. Specify either UNIX or WINDOWS. Set this property according to how you enter paths on your FTP, FTPS, or SFTP server. For example, if you are running an FTP server on Windows but when you log in to the server, you must enter UNIX-style paths (that is, with forward slashes), set this value to UNIX and not WINDOWS. Servers running on Windows often present a UNIX-style file system.
fileEncoding	Required. Defines the character encoding that is used by the file server. This property is used when you transfer files in text mode so that the correct encoding sequences are changed when the files are moved between platforms. For example, UTF-8.
limitedWrite	Optional. The default mode when writing to a file server is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The default is false.
controlEncoding	Optional. The control encoding value for control messages being sent to the protocol file server. This property affects the encoding of the file name that is used and must be compatible with the control encoding of the protocol file server. The default is UTF-8.

General attributes that apply to FTP and FTPS servers only:

Attribute	Description
timeZone	Required. The time zone of the protocol file server that you want to send files to or receive files from. For example: America/New_York or Asia/Tokyo.
locale	Required. The language that is used on the protocol file server that you want to send files to or receive files from. For example: en_US or ja_JP
listFormat	Optional. The listing format that defines the format of the file-listed information that is returned from the protocol file server. Use either Windows or UNIX. The default is UNIX.
listFileRecentDateFormat	Optional. The recent date format (less than a year) for FTP client directory listing on an FTP server. This attribute and the listFileOldDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.

Attribute	Description
listFileOldDateFormat	Optional. The old date format (more than a year) for FTP client directory listing on an FTP server. This attribute and the listFileRecentDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
monthShortNames	Optional. A replacement list of month names that are used to decode date information returned from the protocol file server. This property consists of a list of 12 comma-separated names to override the default locale month values. The default is as defined by the protocol file server.

General attributes that apply to FTP servers only:

Attribute	Description
passiveMode	Optional. Controls whether the connection to the FTP server is passive or active. If you set the value of this property to <code>false</code> , the connection is active. If you set the value to <code>true</code> , the connection is passive. The default is <code>false</code> .

General attributes that apply to FTPS servers only:

Attribute	Description
ftpsType	Optional. Specifies whether the explicit or implicit form of the FTPS protocol is used. The default is <code>explicit</code> .
trustStore	Required. The location of the truststore that is used to determine whether the certificate presented by the FTPS server is trusted.
trustStoreType	Optional. The format of the truststore file. The default is <code>JKS</code> .
keyStore	Optional. The location of the keystore that is used to provide certificate information if challenged by the FTPS server. The default is for the protocol bridge to not be able to connect to FTPS servers that are configured to require the authentication of clients.
keyStoreType	Optional. The format of the keystore file. The default is <code>JKS</code> .
ccc	Optional. Selects whether a clear (unencrypted) command channel is used when authentication has completed. The default value is <code>false</code> , which means that the command channel remains encrypted for the entire duration of the FTPS session. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
protFirst	Optional. Specifies whether the USER/PASS commands are issued to the FTPS server before or after the PBSZ/PROT commands. The default value is <code>false</code> , which means USER/PASS commands are sent first followed by PBSZ/PROT commands. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
auth	Optional. Specifies the protocol that is specified as part of the AUTH command. A specified protocol will be tried first, then the default is to try <code>TLS</code> , <code>SSL</code> , <code>TLS-C</code> , or <code>TLS-P</code> until the FTPS server does not reject with a 504 reply code. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .

<limits>

Container element for attributes that are common to all types of server and for attributes that are specific to a type of server:

General limit attributes that apply to all types of protocol file server:

Attribute	Description
maxListFileNames	Optional. The maximum number of names that are collected when scanning a directory on the protocol file server for file names. The default is 999999999.
maxListDirectoryLevels	Optional. The maximum number of directory levels on the protocol server to recursively scan for file names. The default is 1000.
maxReconnectRetry (This attribute is now deprecated.)	Deprecated. This attribute is not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2, or later. Optional. The maximum number of times a protocol server tries to reconnect before the protocol bridge agent stops trying. The default is 2.
reconnectWaitPeriod (This attribute is now deprecated.)	Deprecated. This attribute is not supported on IBM WebSphere MQ 7.5, or on IBM WebSphere MQ File Transfer Edition 7.0.2 or later. Optional. The time period, in seconds, to wait to before attempting to reconnect. The default is 10 seconds.
maxSessions	Optional. The maximum number of sessions for the protocol server. This number must be greater than or equal to the sum of the maximum number of source and destination transfers for the protocol bridge agent. The default is the sum of the values for the agent properties maxSourceTransfers, maxDestinationTransfers, and maxCommandHandlerThreads, plus 1. If these three properties are using their default values of 25, 25, and 5, the maxSessions default is then 56.
socketTimeout	Optional. The socket timeout in seconds. The value of this attribute is used during file streaming. The default is 30 seconds.

Limit attribute that applies to SFTP servers only:

Attribute	Description
connectionTimeout	Optional. The time, in seconds, to wait for a response from the protocol file server to a connection request. A timeout indicates that the protocol file server is not available. The default value is 30 seconds.
cipherList	Optional. Specifies a comma-separated list of ciphers that are used to communicate between the protocol bridge agent and the SFTP server. The ciphers are called in the order that they are specified in this list. The cipher must be available on the server and the client before it can be used. The ciphers that the protocol bridge agent supports are as follows: <ul style="list-style-type: none"> • blowfish-cbc • 3des-cbc • aes128-cbc • aes192-cbc • aes256-cbc • aes128-ctr • aes192-ctr • aes256-ctr • 3des-ctr

Attribute	Description
	<ul style="list-style-type: none"> • arcfour • arcfour128 • arcfour256 <p>By default, the list of ciphers used by protocol bridge agents is aes128-cbc , aes192-cbc , aes256-cbc.</p>

Connect:Direct credentials file format

The ConnectDirectCredentials.xml file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

The ConnectDirectCredentials.xml file must conform to the ConnectDirectCredentials.xsd schema. The ConnectDirectCredentials.xsd schema document is located in the *MQ_INSTALLATION_PATH/mqft/samples/schema* directory of the MQMFT installation. A sample ConnectDirectCredentials.xml file is located in the *MQ_INSTALLATION_PATH/mqft/samples/credentials* directory of the MQMFT installation.

The file ConnectDirectCredentials.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property **xmlConfigReloadInterval** in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectCredentials.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  This schema defines the format of the XML file that is located in the agent properties
  directory of a Connect:Direct bridge agent. The XML file ConnectDirectCredentials.xml
  is used by the default credential validation of the Connect:Direct bridge.
  For more information, see the WebSphere MQ InfoCenter
-->
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectCredentials"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
  >
  <!--
    <?xml version="1.0" encoding="UTF-8"?>

    <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
      xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
        ConnectDirectCredentials.xsd">
      <tns:agent name="CDAGENT01">
        <tns:pnode name="cdnode*" pattern="wildcard">
          <tns:user name="MUSR_.*"
            ignorecase="true"
            pattern="regex"
            cdUserId="bob"
            cdPassword="passw0rd"
            pnodeUserId="bill"
            pnodePassword="alacazam">
            <tns:snode name="cdnode2" pattern="wildcard" userId="sue" password="foo"/>
          </tns:user>
        </tns:pnode>
      </tns:agent>
    </tns:credentials>

  -->

  <element name="credentials" type="tns:credentialsType"/>

```

```

<complexType name="credentialsType">
  <sequence>
    <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="agentType">
  <sequence>
    <element name="pnode" type="tns:pnodeType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
</complexType>

<complexType name="pnodeType">
  <sequence>
    <element name="user" type="tns:userType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
</complexType>

<complexType name="userType">
  <sequence>
    <element name="snode" type="tns:snodeType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="ignorecase" type="boolean" use="optional"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="cdUserId" type="string" use="optional"/>
  <attribute name="cdUserIdCipher" type="string" use="optional"/>
  <attribute name="cdPassword" type="string" use="optional"/>
  <attribute name="cdPasswordCipher" type="string" use="optional"/>
  <attribute name="pnodeUserId" type="string" use="optional"/>
  <attribute name="pnodeUserIdCipher" type="string" use="optional"/>
  <attribute name="pnodePassword" type="string" use="optional"/>
  <attribute name="pnodePasswordCipher" type="string" use="optional"/>
</complexType>

<complexType name="snodeType">
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="userId" type="string" use="optional"/>
  <attribute name="userIdCipher" type="string" use="optional"/>
  <attribute name="password" type="string" use="optional"/>
  <attribute name="passwordCipher" type="string" use="optional"/>
</complexType>

<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>
</schema>

```

Understanding the ConnectDirectCredentials.xml file

The elements and attributes used in the ConnectDirectCredentials.xml file are described in the following list.

<credentials>

Group element containing elements that describe the credentials used by a Connect:Direct bridge agent to connect to a Connect:Direct node.

<agent>

Group element containing elements for <pnode> definitions for a named agent.

<pnode>

The primary node (PNODE) in the Connect:Direct transfer. This node initiates the connection to the secondary node (SNODE).

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.

Attribute	Description
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> wildcard - wildcards are used regex - Java regular expressions are used

<user>

The IBM MQ user that submits the transfer request.

Attribute	Description
name	The user name that is used with Managed File Transfer. The value of this attribute can be a pattern that matches many user names.
ignorecase	Specifies whether the case of the name is ignored. Valid values for the ignorecase attribute are <ul style="list-style-type: none"> true - the name is not case sensitive false - the name is case sensitive
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> wildcard - wildcards are used regex - Java regular expressions are used
cdUserId or cdUserIdCipher	The user name that is used by the Connect:Direct bridge to connect to its associated Connect:Direct node. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
cdPassword or cdPasswordCipher	The password associated with the user name specified by the cdUserId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
pnodeUserId or pnodeUserIdCipher	The user name that is used by the Connect:Direct primary node. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
pnodePassword or pnodePasswordCipher	The password associated with the user name specified by the pnodeUserId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

<snode>

The Connect:Direct node that performs the role of secondary node (SNODE) during the Connect:Direct file transfer.

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> wildcard - wildcards are used regex - Java regular expressions are used

Attribute	Description
userId or userIdCipher	The user name used to connect to this node during a file transfer. If the fteObfuscate command has been used then the cipher version of the attribute must be used.
password or passwordCipher	The password associated with the user name specified by the userId attribute. If the fteObfuscate command has been used then the cipher version of the attribute must be used.

Example

In this example, the Connect:Direct bridge agent connects to the Connect:Direct node pnode1. When an IBM MQ user with the user name beginning with the prefix fteuser followed by a single character, for example fteuser2, requests a transfer involving the Connect:Direct bridge, the Connect:Direct bridge agent will use the user name cduser and the password passw0rd to connect to the Connect:Direct node pnode1. When the Connect:Direct node pnode1 performs its part of the transfer it uses the user name pnodeuser and the password passw0rd1.

If the secondary node in the Connect:Direct transfer has a name that begins with the prefix FISH, the node pnode1 uses the user name fishuser and the password passw0rd2 to connect to the secondary node. If the secondary node in the Connect:Direct transfer has a name that begins with the prefix CHIPS, the node pnode1 uses the user name chipsuser and the password passw0rd3 to connect to the secondary node.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
ConnectDirectCredentials.xsd">
  <tns:agent name="CDAGENT01">
    <tns:pnode name="pnode1" pattern="wildcard">
      <tns:user name="fteuser?" pattern="wildcard" ignorecase="true"
        cdUserId="cduser" cdPassword="passw0rd"
        pnodeUserId="pnodeuser" pnodePassword="passw0rd1">
      <tns:snode name="FISH*" pattern="wildcard"
        userId="fishuser" password="passw0rd2"/>
      <tns:snode name="CHIPS*" pattern="wildcard"
        userId="chipsuser" password="passw0rd3"/>
    </tns:user>
  </tns:pnode>
</tns:agent>
</tns:credentials>
```

Connect:Direct process definitions file format

The ConnectDirectProcessDefinitions.xml file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

The ConnectDirectProcessDefinitions.xml file must conform to the ConnectDirectProcessDefinitions.xsd schema. The ConnectDirectProcessDefinitions.xsd schema document is located in the *MQ_INSTALLATION_PATH/mqft/samples/schema* directory of the MFT installation. A template ConnectDirectProcessDefinitions.xml file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file ConnectDirectProcessDefinitions.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectProcessDefinitions.xml file.

```
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions">

  <element name="cdprocess" type="tns:cdprocessType"></element>

  <complexType name="cdprocessType">
    <sequence>
      <element name="processSet" type="tns:processSetType"
        minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="processSetType">
    <sequence>
      <element name="condition" type="tns:conditionType"
        minOccurs="0" maxOccurs="1"/>
      <element name="process" type="tns:processType"
        minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>

  <complexType name="conditionType">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element name="match" type="tns:matchType"/>
      <element name="defined" type="tns:definedType"/>
    </choice>
  </complexType>

  <complexType name="matchType">
    <attribute name="variable" type="string" use="required"/>
    <attribute name="value" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
  </complexType>

  <complexType name="definedType">
    <attribute name="variable" type="string" use="required"/>
  </complexType>

  <complexType name="processType">
    <sequence>
      <element name="preTransfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="transfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="postTransferSuccess" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
      <element name="postTransferFailure" type="tns:transferType"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>

  <complexType name="transferType">
    <attribute name="process" type="string" use="required"/>
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex"/>
      <enumeration value="wildcard"/>
    </restriction>
  </simpleType>

</schema>
```

Understanding the ConnectDirectProcessDefinitions.xml file

The elements and attributes used in the ConnectDirectProcessDefinitions.xml file are described in the following list.

cdProcess

The root element of the XML document.

processSet

Group element containing all the information about a set of user-defined processes.

condition

Group element containing the conditions that a transfer is tested against to determine whether the set of processes contained in the processSet element are used.

match

A condition that tests whether a the value of a variable matches a given value.

Attribute	Description
variable	Specifies a variable. The value of this variable is compared with the value of the value attribute. The variable is an intrinsic symbol. For more information, see “Substitution variables for use with user-defined Connect:Direct processes” on page 633.
value	Specifies a pattern to match against the value of the variable specified by the variable attribute.
pattern	Specifies the type of pattern that is used for the value of the value attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> • <code>wildcard</code> - wildcards are used • <code>regex</code> - Java regular expressions are used This attribute is optional and the default is <code>wildcard</code> .

defined

A condition that tests whether a variable has been defined.

Attribute	Description
variable	Specifies a variable. If this variable exists, the match condition is satisfied. The variable is an intrinsic symbol. For more information, see “Substitution variables for use with user-defined Connect:Direct processes” on page 633.

process

Group element containing the information about where to locate the Connect:Direct processes to call when a match is found.

transfer

The Connect:Direct process to call during a transfer request.

Attribute	Description
process	Optional. Specifies the name of a file that contains a Connect:Direct process to call during a transfer request. The file path is relative to the Connect:Direct bridge agent configuration directory. This attribute is optional, the default is to use a process generated by MFT. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see “The use of environment variables in MFT properties” on page 511

Example

In this example, there are three processSet elements.

The first processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*` and a **%FTESUSER** variable with a value of `Admin`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/AdminClient.cdp` as part of the transfer.

The second processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/Client.cdp` as part of the transfer. The Connect:Direct bridge agent reads the processSet elements in the order that they are defined, and if it finds a match, it uses the first match and does not look for another match. For transfer requests that match the conditions of both the first and second processSet, the Connect:Direct bridge agent calls only the processes specified by the first processSet.

The third processSet element has no conditions and matches all transfers. If the transfer request does not match the conditions of the first or second processSet, the Connect:Direct bridge agent submits the Connect:Direct process specified by the third condition. This process is located in the `agent_configuration_directory/Default.cdp` as part of the transfer.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions
ConnectDirectProcessDefinitions.xsd">

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard"/>
      <tns:match variable="%FTESUSER" value="Admin" pattern="wildcard"/>
    </tns:condition>
    <tns:process>
      <tns:transfer process="AdminClient.cdp"/>
    </tns:process>
  </tns:processSet>

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard"/>
    </tns:condition>
    <tns:process>
      <tns:transfer process="Client.cdp"/>
    </tns:process>
  </tns:processSet>

  <tns:processSet>
    <tns:process>
      <tns:transfer process="Default.cdp"/>
    </tns:process>
  </tns:processSet>

</tns:cdprocess>
```

Connect:Direct node properties file format

The `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

The `ConnectDirectNodeProperties.xml` file must conform to the `ConnectDirectNodeProperties.xsd` schema. The `ConnectDirectNodeProperties.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MFT installation. A template `ConnectDirectNodeProperties.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectNodeProperties.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema

The following schema describes which elements are valid in the `ConnectDirectNodeProperties.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
```

```

elementFormDefault="qualified"
xmlns="https://www.w3.org/2001/XMLSchema"
xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties">

<element name="nodeProperties" type="tns:nodePropertiesType"></element>

<complexType name="nodePropertiesType">
  <sequence>
    <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1"/>
    <element name="node" type="tns:nodeType" minOccurs="0" maxOccurs="unbounded"></element>
  </sequence>
</complexType>

<complexType name="nodeType">
  <attribute name="name" type="string" use="required"/>
  <attribute name="pattern" type="tns:patternType" use="optional"/>
  <attribute name="type" type="string" use="required"/>
</complexType>

<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>

</schema>

```

Understanding the ConnectDirectNodeProperties.xml file

The elements and attributes used in the ConnectDirectNodeProperties.xml file are described in the following list.

nodeProperties

Root element of the XML document.

credentialsFile

Path to the credentials file where sensitive information is stored. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see [“The use of environment variables in MFT properties” on page 511](#)

node

Specifies one or more Connect:Direct nodes.

Attribute	Description
name	A pattern that identifies the names of Connect:Direct nodes that use the definitions specified by the node element. Pattern matching is not case sensitive.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are: <ul style="list-style-type: none"> wildcard - wildcards are used regex - Java regular expressions are used For information about the types of regular expressions used by MFT, see “Regular expressions used by MFT” on page 633 .
type	Specifies the operating system type of the Connect:Direct node or nodes that match the pattern given by the name attribute. Valid values for the type attribute are: <ul style="list-style-type: none"> Windows - the node runs on Windows UNIX - the node runs on UNIX or Linux  z/OS, zos, os/390, or os390 - the node runs on z/OS The value of this attribute is not case sensitive.

Example

In this example, the Connect:Direct credentials filename is specified as `ConnectDirectCredentials.xml`. The example code specifies the following platform connections:

- All Connect:Direct nodes that have a name that begins with "cdnodew" run on the Windows platform.
- All Connect:Direct nodes that have a name that begins with "cdnodeu" run on the UNIX platform.
- All Connect:Direct nodes that have a name that begins with "cdnodez" run on the z/OS platform.
- All other Connect:Direct nodes run on the UNIX platform.

The Connect:Direct bridge agent searches for matches from the start of the file to the end, and uses the first match that it finds.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:nodeProperties xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectNodeProperties
    ConnectDirectNodeProperties.xsd">

  <tns:credentialsFile path="ConnectDirectCredentials.xml"/>
  <tns:node name="cdnodew*" pattern="wildcard" type="windows"/>
  <tns:node name="cdnodeu.*" pattern="regex" type="unix"/>
  <tns:node name="cdnodez*" pattern="wildcard" type="zos"
  <tns:node name="*" pattern="wildcard" type="unix"/>

</tns:nodeProperties>
```

fteutils.xsd schema file

This schema defines elements and types used by many of the other Managed File Transfer schemas.

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2008, 2024. All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
-->

<!--
This schema defines elements and types used by many of the other MQMFT schemas.
For more information about MQMFT XML message formats, see
https://www.ibm.com/docs/SSEP7X_7.0.4/com.ibm.wmqfte.doc/message_formats.htm
-->
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <!--
    Defines the version type 1.00 - 99.00
  -->
  <transaction version= 1.00
  -->
  <xsd:simpleType name="versionType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]+\.[0-9][0-9]"/>
    </xsd:restriction>
  </xsd:simpleType>
  <!--
    Defines the transaction reference
  -->
  <transaction version= 1.00 ID="414d5120514d31202020202020202020205ecf0a4920011802"
  -->
  <xsd:simpleType name="IDType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9a-fA-F]{48}"/>
    </xsd:restriction>
```

```

</xsd:simpleType>
<!--
  This is an alias for hostUserIDType.
  Here to allow addition of attributes on originator elements
-->
<xsd:complexType name="origRequestType">
  <xsd:complexContent>
    <xsd:extension base="hostUserIDType">
      <xsd:sequence>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!--
  Defines a Delete originator as a machine and user pair
<hostName>myMachine</hostName>
<userName>myUserId</userName>
-->
<xsd:complexType name="origDeleteType">
  <xsd:sequence>
    <xsd:element name="delete" type="hostUserIDType" maxOccurs="1" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines a machine, user, MQMD userID triple
<hostName>myMachine</hostName>
<userID>myUserId</userID>
<mqmdUserID>MQMDUSERID</mqmdUserID>
-->
<xsd:complexType name="hostUserIDType">
  <xsd:sequence>
    <xsd:element name="hostName" type="xsd:string" minOccurs="0" maxOccurs="1" />
    <xsd:element name="userID" type="xsd:string" minOccurs="0" maxOccurs="1" />
    <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the destinationAgent with agent and queue manager name
<destinationAgent agent="there" QMgr="far" agentType="BRIDGE" bridgeURL="ftp://
server.example.net:21" bridgeNode="DEST_NODE" />
  optional agentType attribute expected to be one of STANDARD, BRIDGE, WEB_GATEWAY,
EMBEDDED, CD_BRIDGE
-->
<xsd:complexType name="agentType">
  <xsd:attribute name="agent" type="xsd:string" use="required" />
  <xsd:attribute name="agentType" type="xsd:string" use="optional" />
  <xsd:attribute name="QMgr" type="xsd:string" use="optional" />
  <xsd:attribute name="bridgeURL" type="xsd:string" use="optional" />
  <xsd:attribute name="bridgeNode" type="xsd:string" use="optional" />
  <xsd:attribute name="pnode" type="xsd:string" use="optional" />
  <xsd:attribute name="snode" type="xsd:string" use="optional" />
</xsd:complexType>
<!--
  Defines the status type; attr/resultCode and 0 or many supplements
  There may also be additional command specific data, either: transfer, ping or call data
<status resultCode="8011">
  <supplement>Azionamento del USB</supplement>
  <supplement>morto come norweign azzurro</supplement>
</status>
-->
<xsd:complexType name="statusType">
  <xsd:sequence>
    <xsd:element name="supplement" type="xsd:string" maxOccurs="unbounded"
minOccurs="0" />
    <xsd:choice>
      <xsd:element name="fileSpace" type="fileSpaceReplyType" minOccurs="0"
maxOccurs="1" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="resultCode" type="resultCodeType" use="required" />
</xsd:complexType>
<!--
  Defines the fileSpace type for use with communication between a web agent
and a web gateway
<fileSpace name="" location=""><Quota bytes="" /></fileSpace>
-->
<xsd:complexType name="fileSpaceReplyType">
  <xsd:attribute name="name" use="required" type="xsd:string" />

```

```

    <xsd:attribute name="location" use="required" type="xsd:string"/>
    <xsd:attribute name="quota" use="required" type="xsd:long"/>
  </xsd:complexType>
  <!--
    Defines the destinationAgent with agent and queue manager name, plus connection
    details.
    <destinationAgent agent="there" QMgr="far"/>
  -->
  <xsd:complexType name="agentClientType">
    <xsd:attribute name="agent" type="xsd:string" use="required"/>
    <xsd:attribute name="QMgr" type="xsd:string" use="optional"/>
    <xsd:attribute name="hostName" type="xsd:string" use="optional"/>
    <xsd:attribute name="portNumber" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:attribute name="channel" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <!--
    Defines the fileURI type as string
    <file encoding="UTF8" EOL="CR">C:/from/here.txt</file>
  -->
  <xsd:complexType name="fileType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="filesystem" type="filesystemNameType" use="optional"/>
        <xsd:attribute name="alias" type="xsd:string" use="optional"/>
        <xsd:attribute name="encoding" type="encodingType" use="optional"/>
        <xsd:attribute name="EOL" type="EOLType" use="optional"/>
        <xsd:attribute name="size" type="xsd:long" use="optional"/>
        <xsd:attribute name="last-modified" type="xsd:dateTime" use="optional"/>
        <xsd:attribute name="delimiter" type="xsd:string" use="optional"/>
        <xsd:attribute name="delimiterType" type="xsd:string" use="optional"/>
        <xsd:attribute name="delimiterPosition" type="delimiterPositionType"
use="optional"/>
        <xsd:attribute name="includeDelimiterInFile" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="keepTrailingSpaces" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="truncateRecords" type="xsd:boolean" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <!--
    Defines the filesystem type as string
    <filesystem>
      <name>tarquin</name>
    </filesystem>
  -->
  <xsd:complexType name="filesystemType">
    <xsd:sequence>
      <xsd:element name="name" type="filesystemNameType"/>
    </xsd:sequence>
  </xsd:complexType>
  <!--
    Defines a name element
    <name>bob</name>
  -->
  <xsd:simpleType name="filesystemNameType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <!--
    Defines the accepted choices for the persistent attribute.
  -->
  <xsd:simpleType name="persistenceType">
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="true"/>
      <xsd:enumeration value="false"/>
      <xsd:enumeration value="qdef"/>
    </xsd:restriction>
  </xsd:simpleType>
  <!--
    Defines the queueURI type as string with all supported attributes.
    <queue>QUEUE@QM</queue>
  -->
  <xsd:complexType name="queueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="persistent" type="persistenceType" use="optional"/>
        <xsd:attribute name="eofMarker" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="setMqProps" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="split" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="useGroups" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="delimiter" type="xsd:string" use="optional"/>
        <xsd:attribute name="delimiterType" type="xsd:string" use="optional"/>
        <xsd:attribute name="delimiterPosition" type="delimiterPositionType"
use="optional"/>

```

```

        <xsd:attribute name="includeDelimiterInMessage" type="xsd:boolean"
use="optional"/>
        <xsd:attribute name="groupId" type="groupIdType" use="optional"/>
        <xsd:attribute name="messageId" type="messageIdType" use="optional"/>
        <xsd:attribute name="messageCount" type="xsd:nonNegativeInteger"
use="optional"/>
        <xsd:attribute name="messageLength" type="xsd:nonNegativeInteger"
use="optional"/>
        <xsd:attribute name="waitTime" type="xsd:nonNegativeInteger" use="optional"/>
        <xsd:attribute name="encoding" type="encodingType" use="optional"/>
        <xsd:attribute name="EOL" type="EOLType" use="optional"/>
        <xsd:attribute name="unrecognisedCodePage" type="unrecognisedCodePageType"
use="optional"/>
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the accepted values for the delimiterPosition attribute.
-->
<xsd:simpleType name="delimiterPositionType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="postfix"/>
        <xsd:enumeration value="prefix"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the groupId type
    <queue groupId="414d5120514d3120202020202020202020205ecf0a4920011802">
    Also allow a substitution variable of the form ${variable}
-->
<xsd:simpleType name="groupIdType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9a-fA-F]{48}|\${.*}" />
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the messageId type
    <queue messageId="414d5120514d3120202020202020202020205ecf0a4920011802">
    Also allow a substitution variable of the form ${variable}
-->
<xsd:simpleType name="messageIdType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9a-fA-F]{48}|\${.*}" />
    </xsd:restriction>
</xsd:simpleType>
<!-- Defines the accepted values for the unrecognisedCodePage attribute. -->
<xsd:simpleType name="unrecognisedCodePageType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="fail"/>
        <xsd:enumeration value="binary"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines a single source file reference
    <source type="file" recursive="false" disposition="leave">
        <file>filename</file>
    </source/>
-->
<xsd:complexType name="fileSourceType">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="file" type="fileType"/>
            <xsd:element name="queue" type="queueType"/>
        </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="type" type="SourceType" use="optional"/>
    <xsd:attribute name="recursive" type="xsd:boolean" use="optional"/>
    <xsd:attribute name="disposition" type="sourceDispositionType" use="optional"/>
    <xsd:attribute name="correlationString1" type="xsd:string" use="optional"/>
    <xsd:attribute name="correlationNum1" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:attribute name="correlationBoolean1" type="xsd:boolean" use="optional"/>
</xsd:complexType>
<!--
    Defines the enumeration values for source type
    type="file|queue"
-->
<xsd:simpleType name="SourceType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="file"/>
        <xsd:enumeration value="directory"/>
        <xsd:enumeration value="queue"/>
        <xsd:enumeration value="dataset"/>
    </xsd:restriction>
</xsd:simpleType>

```

```

        <xsd:enumeration value="pds"/>
        <xsd:enumeration value="filespace"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for source disposition
    disposition="leave|delete"
-->
<xsd:simpleType name="sourceDispositionType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="leave"/>
        <xsd:enumeration value="delete"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines a single destination file reference
    <destination type="file" exist="overwrite">
        <file>filename</file>
    </destination/>
-->
<xsd:complexType name="fileDestinationType">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="file" type="fileType"/>
            <xsd:element name="filespace" type="filespaceType"/>
            <xsd:element name="queue" type="queueType"/>
        </xsd:choice>
        <xsd:element name="attributes" type="attributeType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="DestinationType" use="optional"/>
    <xsd:attribute name="exist" type="existType" use="optional"/>
    <xsd:attribute name="correlationString1" type="xsd:string" use="optional"/>
    <xsd:attribute name="correlationNum1" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:attribute name="correlationBoolean1" type="xsd:boolean" use="optional"/>
</xsd:complexType>
<!--
    Defines the enumeration values for destination file type
    type="file|directory|queue|dataset|pds|filespace"
    'dataset' and 'pds' only apply to z/OS environments.
-->
<xsd:simpleType name="DestinationType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="file"/>
        <xsd:enumeration value="directory"/>
        <xsd:enumeration value="queue"/>
        <xsd:enumeration value="dataset"/>
        <xsd:enumeration value="pds"/>
        <xsd:enumeration value="filespace"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumerations values for file exists on destination behavior
    exist="error|overwrite"
-->
<xsd:simpleType name="existType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="error"/>
        <xsd:enumeration value="overwrite"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines one or more file attributes
    <destination encoding=? CFLF=?>
        <file>filename</file>
        <attributes>
            <attribute>DIST(MIRRORED,UPDATE)</attribute>
        </attributes>
    </destination/>
-->
<xsd:complexType name="attributeType">
    <xsd:sequence>
        <xsd:element name="attribute" type="xsd:string" maxOccurs="unbounded"
minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines a single file reference
    <source encodings=? CFLF=?>
        <file>filename</file>

```

```

        <checksum method="MD5">3445678</checksum>
    </source/>
    .. OR ..
    <destination encoding=? CFLF=?>
        <file>filename</file>
        <checksum method="MD5">3445678</checksum>
    </destination/>
-->
<xsd:complexType name="fileCheckSumType">
    <xsd:sequence>
        <xsd:element name="file" type="fileType"/>
        <xsd:element name="checksum" type="checksumType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines the checksum type and method
    <checksum method="MD5|none">3445678</checksum>
-->
<xsd:complexType name="checksumType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="method" type="checkSumMethod" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the enumeration values for checkSumMethod
    <checksum method="MD5|none">3445678</checksum>
    Note: uppercase is used since MD5 is acronym and normally written uppercase.
-->
<xsd:simpleType name="checkSumMethod">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="none"/>
        <xsd:enumeration value="MD5"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for agentRole
    agentRole="sourceAgent|destinationAgent"
-->
<xsd:simpleType name="agentRoleType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="sourceAgent"/>
        <xsd:enumeration value="destinationAgent"/>
        <xsd:enumeration value="callAgent"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for mode.
    text, binary or a substitution variable
    <item mode="binary|text|${variableName}">
-->
<xsd:simpleType name="modeType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="binary|text|$\{.*\}"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for EOL
    <file EOL="LF|CRLF">
-->
<xsd:simpleType name="EOLType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="LF"/>
        <xsd:enumeration value="CRLF"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the encoding type as a string
-->
<xsd:simpleType name="encodingType">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!--
    <schedule>
        <submit timebase="source"|"admin">2008-12-07T16:07</submit>
        <repeat>
            <frequency interval="hours">2</frequency>
            <expireTime>2008-12-0816:07</exipreTime>
        </repeat>
    </schedule>
-->

```

```

<xsd:complexType name="scheduleType">
  <xsd:sequence>
    <xsd:element name="submit" type="submitType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="repeat" type="repeatType" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  -->
  <submit timebase="source|admin|UTC">2008-12-07T16:07</submit>
-->
<xsd:complexType name="submitType">
  <xsd:simpleContent>
    <xsd:extension base="noZoneTimeType">
      <xsd:attribute name="timebase" type="timebaseType" use="required"/>
      <xsd:attribute name="timezone" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  <repeat>
    <frequency interval="hours">2</frequency>
    ..optionally..
    <expireTime>2008-12-0816:07</expireTime>
    ..or..
    <expireCount>2</expireCount>
  </repeat>
-->
<xsd:complexType name="repeatType">
  <xsd:sequence>
    <xsd:element name="frequency" type="freqType" maxOccurs="1" minOccurs="1"/>
    <xsd:choice minOccurs="0">
      <xsd:element name="expireTime" type="noZoneTimeType"/>
      <xsd:element name="expireCount" type="positiveIntegerType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<!--
  <frequency interval="hours">2</frequency>
-->
<xsd:complexType name="freqType">
  <xsd:simpleContent>
    <xsd:extension base="positiveIntegerType">
      <xsd:attribute name="interval" type="intervalType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines positive integer type
  i.e., 1+
-->
<xsd:simpleType name="positiveIntegerType">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the interval enumeration values of
  "minutes", "hours", "days", "weeks", "months" or "years"
-->
<xsd:simpleType name="intervalType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="minutes"/>
    <xsd:enumeration value="hours"/>
    <xsd:enumeration value="days"/>
    <xsd:enumeration value="weeks"/>
    <xsd:enumeration value="months"/>
    <xsd:enumeration value="years"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the interval of either "source", "admin" or "UTC"
  source = use timezone of the source Agent.
  admin = use timezone of the administrator executing the command script.
  UTC = Timezone is UTC.
-->
<xsd:simpleType name="timebaseType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="source"/>
    <xsd:enumeration value="admin"/>
    <xsd:enumeration value="UTC"/>
  </xsd:restriction>
</xsd:simpleType>
<!--

```

```

    Defines a date and time without a time zone (2008-12-08T16:07)
-->
<xsd:simpleType name="noZoneTimeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[\n\r\t ]*\d{4}\-(0[1-9]|1[0-2])\-(0[1-9]|[1-2][0-9]|
3[0-1])T([0-1][0-9]|2[0-3]):[0-5][0-9]([\+\-]\d{4}|Z)?[\n\r\t ]*" />
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the ID element, e.g. 56
-->
<xsd:simpleType name="idType">
  <xsd:restriction base="xsd:string" />
</xsd:simpleType>
<!--
  Defines the resultCode type -2 - 9999
  <status resultCode="8011">
-->
<xsd:simpleType name="resultCodeType">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="-2" />
    <xsd:maxInclusive value="9999" />
  </xsd:restriction>
</xsd:simpleType>
<!--
  Define the metaDataSet type comprising one or more key value pairs
  <metaDataSet>
    <metaData key="name">value</metaData>
    <metaData key="name">value</metaData>
  </metaDataSet>
-->
<xsd:complexType name="metaDataSetType">
  <xsd:sequence>
    <xsd:element name="metaData" type="metaDataType" maxOccurs="unbounded"
minOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the metaData type which is made up of a key and a value
  <metaData key="name">value</metaData>
-->
<xsd:complexType name="metaDataType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="key" type="xsd:string" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines containing element for triggers
  <trigger log="yes">
    <fileExist comparison="=" value="Exist">file1</fileExist>
    <fileSize comparison=">=" value="1GB">file1</fileSize>
  </trigger>
-->
<xsd:complexType name="triggerType">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="fileExist" type="fileExistTriggerType" maxOccurs="unbounded"
minOccurs="1" />
    <xsd:element name="fileSize" type="fileSizeTriggerType" maxOccurs="unbounded"
minOccurs="1" />
  </xsd:choice>
  <xsd:attribute name="log" type="logEnabledType" use="required" />
</xsd:complexType>
<!--
  Defines the file exists trigger type
  <fileExist comparison="=" value="Exist">file1</trigger>
-->
<xsd:complexType name="fileExistTriggerType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="comparison" type="comparisonFileExistTriggerType"
use="required" />
      <xsd:attribute name="value" type="valueFileExistTriggerType" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines file size trigger type
  <fileSize comparison="=" value="1GB">file1,file2,file3</trigger>
-->
<xsd:complexType name="fileSizeTriggerType">

```

```

        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="comparison" type="comparisonFileSizeTriggerType"
use="required"/>
                <xsd:attribute name="value" type="valueFileSizeTriggerType" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!--
        Defines the enumeration values for file exists trigger conditions
        valueFileExistTriggerType="exist|noexist"
    -->
    <xsd:simpleType name="valueFileExistTriggerType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="exist"/>
            <xsd:enumeration value="noexist"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--
        Defines the enumeration values for file exists trigger comparison operator
        comparisonFileExistTriggerType="="|"!="
    -->
    <xsd:simpleType name="comparisonFileExistTriggerType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="="/>
            <xsd:enumeration value="!="/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--
        Defines the enumeration values for file size trigger comparison operator
        comparisonFileSizeTriggerType=">="
    -->
    <xsd:simpleType name="comparisonFileSizeTriggerType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="&gt;="/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--
        Defines the file size value pattern
        <fileSize comparison=">=" value="10|10B|10KB|10MB|10GB">file1</fileSize>
    -->
    <xsd:simpleType name="valueFileSizeTriggerType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0123456789]+([bB]|[kK][bB]|[mM][bB]|[gG][bB]|)"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--
        Defines the enumeration values for trigger logging enabled flag
        <trigger log="yes|no">
    -->
    <xsd:simpleType name="logEnabledType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="yes"/>
            <xsd:enumeration value="no"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--
        Defines the reply type
        <reply QMGR="QMGR name" persistent="true">Queue Name</reply>
    -->
    <xsd:complexType name="replyType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="QMGR" type="xsd:string" use="required"/>
                <xsd:attribute name="persistent" type="persistenceType" use="optional"/>
                <xsd:attribute name="detailed" type="detailedType"
use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <!--
        Defines the accepted choices for the detailed attribute.
    -->
    <xsd:simpleType name="detailedType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="true"/>
            <xsd:enumeration value="false"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!--

```

```

        Defines the priority type
        <transferset priority="1">
-->
<xsd:simpleType name="priorityType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0123456789]"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Define the job information element
  <job>
    <name>JOBNAME</name>
  </job>
-->
<xsd:complexType name="jobType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines an action
  <action>
    <runCommand name="myCommand.sh"/>
  </action>
-->
<xsd:complexType name="commandActionType">
  <xsd:choice>
    <xsd:element name="command" type="commandType" maxOccurs="1" minOccurs="0"/>
  </xsd:choice>
</xsd:complexType>
<!--
  Defines a command
  <command name="runme" successRC="0" maxReplyLength="1024">
    <argument>firstArg</argument>
    <argument>secondArg</argument>
  </command>
-->
<xsd:complexType name="commandType">
  <xsd:sequence>
    <xsd:element name="argument" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="target" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="property" type="propertyType" maxOccurs="unbounded"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="successRC" type="xsd:string" use="optional"/>
  <xsd:attribute name="retryCount" type="nonNegativeIntegerType" use="optional"/>
  <xsd:attribute name="retryWait" type="nonNegativeIntegerType" use="optional"/>
  <xsd:attribute name="type" type="callTypeType" use="optional"/>
  <xsd:attribute name="priority" type="commandPriorityType" use="optional"/>
  <xsd:attribute name="message" type="xsd:string" use="optional"/>
</xsd:complexType>
<!--
  Defines the enumeration values for the type of a command
  type="executable|antscript|jcl"
-->
<xsd:simpleType name="callTypeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="executable"/>
    <xsd:enumeration value="antscript"/>
    <xsd:enumeration value="jcl"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the priority type for a command
  priority="5"
-->
<xsd:simpleType name="commandPriorityType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[123456789]"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the property type that is used as a child of commandType
  <property name="xxx" value="yyy"/>
-->
<xsd:complexType name="propertyType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
<!-- Defines a non-negative integer type -->

```

```

<xsd:simpleType name="nonNegativeIntegerType">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the transfer command specific reply information, to be included as part the
general reply
  <transferReply>
    <preSourceData>
      <runCommandReply resultCode="0">
        <stdout>
          <line>the quick brown fox jumped over the lazy dog</line>
        </stdout>
        <stderr></stderr>
      </runCommandReply>
    </preSourceData>
  </transferReply>
-->
<xsd:complexType name="transferReplyType">
  <xsd:sequence>
    <xsd:element name="preSourceData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="postSourceData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="preDestinationData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="postDestinationData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the action reply type information
  <actionReply>
    <runCommandReply resultCode="1">
      <stdout></stdout>
      <stderr>
        <line>permission denied</line>
      </stderr>
    </runCommandReply>
  </actionReply>
-->
<xsd:complexType name="actionReplyType">
  <xsd:choice>
    <xsd:element name="runCommandReply" type="commandReplyType" maxOccurs="1"
minOccurs="0"/>
  </xsd:choice>
</xsd:complexType>
<!--
  Defines command specific reply information, to be included as part the general reply
  <commandReply resultCode="0">
    <stdout>
      <line>first line of output text</line>
      <line>second line of output text</line>
    </stdout>
    <stderr>
      <line>line of error text</line>
    </stderr>
  </commandReply>
-->
<xsd:complexType name="commandReplyType">
  <xsd:sequence>
    <xsd:element name="stdout" type="textLinesType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="stderr" type="textLinesType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="resultCode" type="xsd:int" use="required"/>
</xsd:complexType>
<!-- Defines type for lines of text -->
<xsd:complexType name="textLinesType">
  <xsd:sequence>
    <xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines the ping agent command specific reply information, to be included as part the
general reply
  <pingAgentReply resultCode="0">
    <agentVersion>Build level: f000-20090408-1200</agentVersion>
  </pingAgentReply>
-->
<xsd:complexType name="pingAgentReplyType">
  <xsd:sequence>

```

```

        <xsd:element name="agentVersion" type="xsd:string" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines sequence of exit elements
    <exit ...
    <exit ...
    -->
<xsd:complexType name="exitGroupType">
    <xsd:sequence>
        <xsd:element name="exit" type="exitType" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines the outcome of calling a command
    <command ...
    <callResult ...
    -->
<xsd:complexType name="callGroupType">
    <xsd:sequence>
        <xsd:element name="command" type="commandType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="callResult" type="callResultType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines either the successful call of a command, or a failed attempt to call a command
    <callResultType outcome="success|failure|error" retries="X">
        <result ... />
    </callResultType>
    -->
<xsd:complexType name="callResultType">
    <xsd:sequence>
        <xsd:element name="result" type="resultType" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="returnCode" type="xsd:integer" use="optional"/>
    <xsd:attribute name="retries" type="xsd:integer" use="optional"/>
    <xsd:attribute name="outcome" type="outcomeType" use="required"/>
</xsd:complexType>
<!--
    Defines the information recorded for the successful call of a command
    <result...>
        <stdout...
        <stderr...
        <error...
    </result...>
    -->
<xsd:complexType name="resultType">
    <xsd:sequence>
        <xsd:element name="stdout" type="outputType" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="stderr" type="outputType" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="error" type="xsd:string" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="returnCode" type="xsd:integer" use="optional"/>
    <xsd:attribute name="outcome" type="outcomeType" use="required"/>
    <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
</xsd:complexType>
<!-- Enumeration of call outcomes - success, failure or error -->
<xsd:simpleType name="outcomeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="success"/>
        <xsd:enumeration value="failure"/>
        <xsd:enumeration value="error"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the information recorded for each line of standard output / standard error
    generated by calling a program
    <line>line 1</line>
    <line>line 2</line>
    etc.
    -->
<xsd:complexType name="outputType">
    <xsd:sequence>
        <xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines the information recorded for an unsuccessful program call.
    -->
<xsd:complexType name="callFailedType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string"/>
    </xsd:simpleContent>
</xsd:complexType>

```

```

    </xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the exit type; records the transfer exit class name and a status message
    <exit name="class com.example.exit.StartExit">
        <status ...
    </exit>
-->
<xsd:complexType name="exitType">
    <xsd:sequence>
        <xsd:element name="status" type="exitStatusType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
<!--
    Defines exit status to record whether exit voted to proceed or cancel transfer.
    <status resultCode="proceed">
        <supplement>go ahead</supplement>
    </status>
-->
<xsd:complexType name="exitStatusType">
    <xsd:sequence>
        <xsd:element name="supplement" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="resultCode" type="exitResultEnumType" use="optional"/>
</xsd:complexType>
<!--
    Defines the enumeration for transfer exit result values.
    <status resultCode="proceed">
-->
<xsd:simpleType name="exitResultEnumType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="proceed"/>
        <xsd:enumeration value="cancelTransfer"/>
        <xsd:enumeration value="cancelTask"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Note: From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.

Related concepts

[“XML message formats used by MFT” on page 708](#)

Managed File Transfer uses messages in XML format for a number of purposes: to command an agent; to log information about the monitors, schedules, and transfers; and to define information used for configuration. The logical structure of the XML formats used for these purposes described by XML schema.

MFT developing applications reference

Running programs before or after a file transfer

Examples of using `fteCreateTransfer` to start programs

You can use the **`fteCreateTransfer`** command to specify programs to run before or after a transfer.

In addition to using **`fteCreateTransfer`**, there are other ways to invoke a program before or after a transfer. For more information, see [“Specifying programs to run with MFT” on page 239](#).

All these examples use the following syntax to specify a program:

```
[type:]commandspec[, [retrycount][, [retrywait][, successrc]]]
```

For more information about this syntax, see [“`fteCreateTransfer`: start a new file transfer” on page 417](#).

Running an executable program

The following example specifies an executable program called `mycommand` and passes two arguments, `a` and `b`, to the program.

```
mycommand(a,b)
```

To run this program at the source agent `AGENT1` before the transfer starts, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Running, and retrying, an executable program

The following example specifies an executable program called `simple`, which does not take any arguments. A value of `1` is specified for `retrycount` and a value of `5` is specified for `retrywait`. These values mean that the program will be retried once if it does not return a successful return code, after a wait of five seconds. No value is specified for `successrc`, so the only successful return code is the default value of `0`.

```
executable:simple,1,5
```

To run this program at the source agent `AGENT1` after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Running an Ant script and specifying successful return codes

The following example specifies an Ant script called `myscript` and passes two properties to the script. The script is run using the **`fteAnt`** command. The value for `successrc` is specified as `>2<7&!5|0|14`, which specifies that return codes of `0`, `3`, `4`, `6`, and `14` indicate success.

```
antscript:myscript(prop1=fred,prop2=bob),,,>2<7&!5|0|14
```

To run this program at the destination agent `AGENT2` before the transfer has started, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2<7&!5|0|14"destinationSpecification sourceSpecification
```

Running an Ant script and specifying targets to call

The following example specifies an Ant script called `script2` and two targets, `target1` and `target2`, to call. The property `prop1` is also passed in, with a value of `recmfm(F,B)`. The comma (`,`) and parentheses in this value are escaped using a backslash character (`\`).

```
antscript:script2(target1,target2,prop1=recmfm\F\B\),,,>2<7&!5|0|14
```

To run this program at the destination agent `AGENT2` after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmfm\F\B\),,,>2<7&!5|0|14"
destinationSpecification sourceSpecification
```

Using metadata in an Ant script

You can specify an Ant task as any of the following calls for a transfer:

- pre source
- post source
- predestination
- post destination

When the Ant task is run, the user meta data of the transfer is made available using environment variables. You can access this data using, for example, the following code:

```
<property environment="environment"/>
<echo>${environment.mymetadata}</echo>
```

where mymetadata is the name of some meta data inserted into the transfer.

Running a JCL script

The following example specifies a JCL script called ZOSBATCH. A value of 3 is specified for `retrycount`, a value of 30 is specified for `retrywait` and a value of 0 is specified for `successrc`. These values mean that the script is retried three times if it does not return a successful return code of 0, with a wait of thirty seconds between each attempt.

```
jcl:ZOSBATCH,3,30,0
```

where ZOSBATCH is a member of a PDS called MYSYS.JCL, and the `agent.properties` file contains the line `commandPath=...:/'MYSYS.JCL':...`

To run this program at the source agent AGENT1 after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0
destinationSpecification sourceSpecification
```

Related concepts

[“Specifying programs to run with MFT” on page 239](#)

You can run programs on a system where a Managed File Transfer Agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference

[“fteCreateTransfer: start a new file transfer” on page 417](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Using Apache Ant with MFT

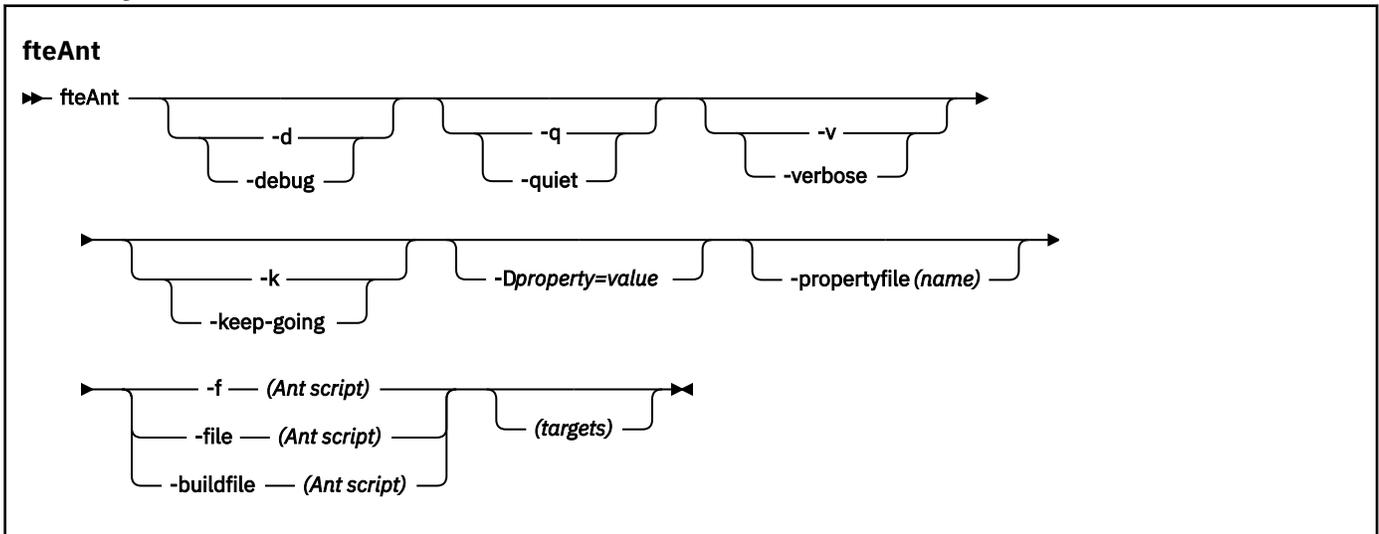
fteAnt: run Ant tasks in MFT

The **fteAnt** command runs Ant scripts in an environment that has Managed File Transfer Ant tasks available.

Purpose

Use the **fteAnt** command to run an Ant script in an environment with Managed File Transfer. Unlike the standard **ant** command, **fteAnt** requires that you define a script file.

Syntax



Parameters

-debug or -d

Optional. Generate debugging output.

-quiet or -q

Optional. Generate minimal output.

-verbose or -v

Optional. Generate verbose output.

-keep-going or -k

Optional. Execute all targets that do not depend on failed targets.

-D property=value

Optional. Use *value* for a given *property*. Properties that are set with **-D** take precedence over those set in a properties file.

Use the property `com.ibm.wmqfte.propertyset` to specify the set of configuration options that are used for Ant tasks. Use the name of a non-default coordination queue manager as the value for this property. Ant tasks then use the set of configuration options that are associated with this non-default coordination queue manager. If you do not specify this property, the default set of configuration options that are based on the default coordination queue manager is used. If you specify the `cmdqm` attribute for an Ant task, this attribute takes precedence over the set of configuration options that are specified for the `fteAnt` command. This behavior applies regardless of whether you are using the default set of configuration options or specifying a set with the `com.ibm.wmqfte.propertyset` property.

-propertyfile (name)

Optional. Load all properties from a file with **-D** properties taking precedence.

-f (Ant script), -file (Ant script), or -buildfile (Ant script)

Required. Specifies the name of the Ant script to run.

targets

Optional. The name of one or more targets to run from the Ant script. If you do not specify a value for this parameter, the default target for the script is run.

-version

Optional. Displays the Managed File Transfer command and Ant versions.

-? or -h

Optional. Displays command syntax.

Example

In this example, the target **copy** in Ant script `fte_script.xml` is run and the command writes debugging output to standard out.

```
fteAnt -d -f fte_script.xml copy
```

Return codes

0

Command completed successfully.

1

Command ended unsuccessfully.

Other status return codes can also be specified from Ant scripts, for example by using the Ant fail task.

See [Fail](#) for more information.

Ant tasks provided by MFT

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Tasks

- [“fte:awaitoutcome Ant task” on page 822](#)
- [fte:call](#)
- [fte:cancel](#)
- [fte:filecopy](#)
- [fte:filemove](#)
- [fte:ignoreoutcome](#)
- [fte:ping](#)
- [fte:uuid](#)

Nested parameters

The following nested parameters describe nested sets of elements, which are common across several of the supplied Ant tasks:

- [fte:filespec](#)
- [fte:metadata](#)
- [Parameters for program invocation](#)

fte:awaitoutcome Ant task

Waits for a **fte:filecopy**, **fte:filemove**, or **fte:call** operation to complete.

Attributes

id

Required. Identifies the transfer to await an outcome from. Typically, this is a property set by the `idProperty` attribute of the [fte:filecopy](#), [fte:filemove](#), or [fte:call](#) tasks.

rcproperty

Required. Names a property to store the return code of the **fte:awaitoutcome** task in.

timeout

Optional. The maximum amount of time, in seconds, to wait for the operation to complete. The minimum timeout is one second. If you do not specify a timeout value, the **fte:awaitoutcome** task waits forever for the outcome of the operation to be determined.

Example

In this example a file copy is started, and its identifier is stored in the `copy.id` property. While the copy is progressing, other processing can take place. The **fte:awaitoutcome** statement is used to wait until the copy operation completes. The **fte:awaitoutcome** statement identifies which operation to wait for using the identifier stored in the `copy.id` property. The **fte:awaitoutcome** stores a return code indicating the outcome of the copy operation into a property called `copy.result`.

```
<-- issue a file copy request -->
<fte:filecopy
  src="AGENT1@QM1"
  dst="AGENT2@QM2"
  idproperty="copy.id"
  outcome="defer">

  <fte:filespec
    srcfilespec="/home/fteuser1/file.bin"
    dstdir="/home/fteuser2"/>

</fte:filecopy>

<fte:awaitoutcome id="{copy.id}" rcProperty="copy.rc"/>

<echo>Copy id="{copy.id}" rc="{copy.rc}</echo>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:call Ant task

You can use the **fte:call** task to remotely call scripts and programs.

This task allows you to send a **fte:call** request to an agent. The agent processes this request by running a script or program and returning the outcome. The commands to call must be accessible to the agent. Ensure the `commandPath` property value in the `agent.properties` file includes the location of the commands to call. Any path information specified by the `command` nested element must be relative to the locations specified by the `commandPath` property. By default `commandPath` is empty so that the agent cannot call any commands. For more information about this property, see [“commandPath MFT property” on page 347](#).

For more information about the `agent.properties` file, see [“The MFT agent.properties file” on page 526](#).

Attributes

agent

Required. Specifies the agent to submit the **fte:call** request to. Specify the agent information in the form: `agentname@qmgrname` where `agentname` is the name of the agent and `qmgrname` is the name of the queue manager that this agent is directly connected to.

cmdqm

Optional. The command queue manager to submit the request to. Specify this information in the form `qmgrname@host@port@channel`, where:

- `qmgrname` is the name of the queue manager
- `host` is the optional host name of the system where the queue manager is running
- `port` is the optional port number that the queue manager is listening on
- `channel` is the optional SVRCONN channel to use

If you omit the `host`, `port`, or `channel` information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see [“The MFT command.properties file”](#) on page 522.

You can use the `com.ibm.wmqfte.propertySet` property to specify which `command.properties` file to use. For more information, see [com.ibm.wmqfte.propertySet](#).

If you do not use the `cmdqm` attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the `cmdqm` attribute, that is, `qmgrname@host@port@channel`.

idproperty

Optional unless you have specified an outcome of `defer`. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.

You cannot specify this property if you have also specified an outcome property of `ignore`. However, you must specify `idproperty` if you have also specified an outcome property of `defer`.

jobname

Optional. Assigns a job name to the `fte:call` request. You can use job names to create logical groups of transfers. Use the [“fte:uuid Ant task”](#) on page 835 task to generate pseudo-unique job names. If you do not use the `jobname` attribute, the task defaults to using the `com.ibm.wmqfte.ant.jobName` property value, if this property is set. If you do not set this property, no job name is associated with the `fte:call` request.

origuser

Optional. Specifies the originating user identifier to associate with the `fte:call` request. If you do not use the `origuser` attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome

Optional. Determines whether the task waits for the `fte:call` operation to complete before returning control to the Ant script. Specify one of the following options:

await

The task waits for the `fte:call` operation to complete before returning. When an outcome of `await` is specified the `idproperty` attribute is optional.

defer

The task returns as soon as the `fte:call` request has been submitted and assumes that the outcome of the call operation is dealt with later using either the [awaitoutcome](#) or [ignoreoutcome](#) tasks. When an outcome of `defer` is specified the `idproperty` attribute is required.

ignore

If the outcome of the `fte:call` operation is not important, you can specify a value of `ignore`. The task then returns as soon as the `fte:call` request has been submitted, without allocating any resources for tracking the outcome of the command. When an outcome of `ignore` is specified the `idproperty` attribute cannot be specified.

If you do not specify the outcome attribute, the task defaults to using the value `await`.

rcproperty

Optional. Specifies the name of a property to assign the result code of the **fte:call** request to. The result code reflects the overall outcome of the **fte:call** request.

You cannot specify this property if you have also specified an outcome property of `ignore` or `defer`. However, you must specify `rcproperty` if you have specified an outcome of `await`.

Parameters specified as nested elements

fte:command

Specifies the command to be called by the agent. You can only associate a single `fte:command` element with a given **fte:call** operation. The command to be called must be located on the path specified by the `commandPath` property in the agent's `agent.properties` file.

fte:metadata

You can specify metadata to associate with the call operation. This metadata is recorded in the log messages generated by the call operation. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata.

Example

This example shows how to call a command at AGENT1 running on queue manager QM1. The command to call is the script `command.sh`, and the script is called with a single argument of `xyz`. The command `command.sh` is located on the path specified by the `commandPath` property in the agent's `agent.properties` file.

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="AGENT1@QM1"
  rcproperty="call.rc"
  origuser="bob"
  jobname="{job.id}">
  <fte:command command="command.sh" successsrc="1" retrycount="5" retrywait="30">
    <fte:arg value="xyz"/>
  </fte:command>
  <fte:metadata>
    <fte:entry name="org.foo.accountName" value="BDG3R"/>
  </fte:metadata>
</fte:call>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:cancel Ant task

Cancels a Managed File Transfer managed transfer or managed call. A managed transfer might have been created using the **fte:filecopy** or **fte:filemove** tasks. A managed call might have been created using the **fte:call** task.

Attributes

agent

Required. Specifies the agent to submit the **fte:cancel** request to. The value is in the form: *agentname@qmgrname* where *agentname* is the name of the agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

cmdqm

Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host*, *port*, or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see [“The MFT command.properties file” on page 522](#).

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see [com.ibm.wmqfte.propertySet](#).

If you do not use the `cmdqm` attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the `cmdqm` attribute, that is, `qmgrname@host@port@channel`.

id

Required. Specifies the transfer identifier of the transfer to cancel. Transfer identifiers are generated at the point a transfer request is submitted by both the [fte:filecopy](#) and [fte:filemove](#) tasks.

origuser

Optional. Specifies the originating user identifier to associate with the **cancel** request. If the `origuser` attribute is not used, the task defaults to using the user ID that is used to run the Ant script.

Example

The example sends a **fte:cancel** request to the command queue manager `qm0`. The **fte:cancel** request is targeted at `agent1` on queue manager `qm1` for the transfer identifier populated by the `transfer.id` variable. The request is run using the "bob" user ID.

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="agent1@qm1"
  id="{transfer.id}"
  origuser="bob"/>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filecopy Ant task

The **fte:filecopy** task copies files between Managed File Transfer agents. The file is not deleted from the source agent.

Attributes

cmdqm

Optional. The command queue manager to submit the request to. Specify this information in the form `qmgrname@host@port@channel`, where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host*, *port*, or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see [“The MFT `command.properties` file” on page 522](#).

You can use the **`com.ibm.wmqfte.propertySet`** property to specify which `command.properties` file to use. For more information, see [`com.ibm.wmqfte.propertySet`](#).

If you do not use the `cmdqm` attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the `cmdqm` attribute, that is, `qmgrname@host@port@channel`.

dst

Required. Specifies the destination agent for the copy operation. Specify this information in the form: `agentname@qmgrname` where `agentname` is the name of the destination agent and `qmgrname` is the name of the queue manager that this agent is directly connected to.

idproperty

Optional unless you have specified an outcome of `defer`. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.

You cannot specify this property if you have also specified an outcome property of `ignore`. However, you must specify `idproperty` if you have also specified an outcome property of `defer`.

jobname

Optional. Assigns a job name to the copy request. You can use job names to create logical groups of transfers. Use the [“`fte:uuid Ant task`” on page 835](#) task to generate pseudo-unique job names. If you do not use the `jobname` attribute, the task defaults to using the `com.ibm.wmqfte.ant.jobName` property value, if this property is set. If you do not set this property, no job name is associated with the copy request.

origuser

Optional. Specifies the originating user identifier to associate with the copy request. If you do not use the `origuser` attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome

Optional. Determines whether the task waits for the copy operation to complete before returning control to the Ant script. Specify one of the following options:

await

The task waits for the copy operation to complete before returning. When an outcome of `await` is specified the `idproperty` attribute is optional.

defer

The task returns as soon as the copy request has been submitted and assumes that the outcome of the copy operation is dealt with later using either the [`awaitoutcome`](#) or [“`fte:ignoreoutcome Ant task`” on page 833](#) tasks. When an outcome of `defer` is specified the `idproperty` attribute is required.

ignore

If the outcome of the copy operation is not important, you can specify a value of `ignore`. The task then returns as soon as the copy request has been submitted, without allocating any resources for tracking the outcome of the transfer. When an outcome of `ignore` is specified the `idproperty` attribute cannot be specified.

If you do not specify the outcome attribute, the task defaults to using the value `await`.

priority

Optional. Specifies the priority to associate with the copy request. In general, higher priority transfer requests take precedence over lower priority requests. The priority value must be in the range 0 - 9 (inclusive). A priority value of 0 is the lowest priority and a value of 9 is the highest priority. If you do not specify the `priority` attribute, the transfer defaults to a priority of 0.

rcproperty

Optional. Specifies the name of a property to assign the result code of the copy request to. The result code reflects the overall outcome of the copy request.

You cannot specify this property if you have also specified an outcome property of `ignore` or `defer`. However, you must specify `rcproperty` if you specify an outcome of `await`.

V 9.0.1 transferRecoveryTimeout

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

-1

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

0

The agent stops the file transfer as soon as it enters recovery.

>0

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result" transferRecoveryTimeout="21600">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
  file.bin"/>
</fte:filecopy>
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this attribute is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a property to the [The agent.properties file](#). For more information, see [“Timeout option for file transfers in recovery” on page 236](#).

src

Required. Specifies the source agent for the copy operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the source agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

Parameters specified as nested elements

fte:filespec

Required. You must specify at least one file specification that identifies the files to copy. You can specify more than one file specification if required. See the [fte:filespec](#) topic for more information.

fte:metadata

You can specify metadata to associate with the copy operation. This metadata is carried with the transfer and is recorded in the log messages generated by the transfer. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata. See the [fte:metadata](#) topic for more information.

fte:presrc

Specifies a program invocation to take place at the source agent before the transfer starts. You can only associate a single `fte:presrc` element with a given transfer. See the [program invocation](#) topic for more information.

fte:predst

Specifies a program invocation to take place at the destination agent before the transfer starts. You can only associate a single `fte:predst` element with a given transfer. See the [program invocation](#) topic for more information.

fte:postsrc

Specifies a program invocation to take place at the source agent after the transfer has completed. You can only associate a single `fte:postsrc` element with a given transfer. See the [program invocation](#) topic for more information.

fte:postdst

Specifies a program invocation to take place at the destination agent after the transfer has completed. You can only associate a single `fte:postdst` element with a given transfer. See the [program invocation](#) topic for more information.

If `fte:presrc`, `fte:predst`, `fte:postsrc`, `fte:postdst`, and `exits` do not return a success status, the rules are as follows in the order specified:

1. Run the source start exits. If source start exits fail the transfer fails and nothing further is run.
2. Run the pre-source call (when present). If the pre-source call fails, the transfer fails and nothing further is run.
3. Run the destination start exits. If the destination start exits fail the transfer fails and nothing further is run.
4. Run the pre-destination call (when present). If the pre-destination call fails, the transfer fails and nothing further is run.
5. Perform the file transfers.
6. Run the destination end exits. There is no failure status for these exits.
7. If the transfer is successful (if some files transfer successfully, it is deemed successful) run the post-destination call (if present). If the post-destination call fails, the transfer fails.
8. Run the source end exits. There is no failure status for these exits.
9. If the transfer is successful run the post-source call (if present). If the post-source call fails, the transfer fails.

Examples

This example shows a basic file transfer between `agent1` and `agent2`. The command to start the file transfer is sent to a queue manager called `qm0`, using a client transport mode connection. The result of the file transfer operation is assigned to the property called `copy.result`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
</fte:filecopy>
```

This example shows the same file transfer, but with the addition of metadata and a program start to take place at the source agent after the transfer has completed.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">
  <fte:metadata>
```

```

        <fte:entry name="org.example.departId" value="ACCOUNTS"/>
        <fte:entry name="org.example.batchGroup" value="A1"/>
    </fte:metadata>

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

        <fte:postsrc command="/home/fteuser2/scripts/post.sh" successsrc="1" >
            <fte:arg value="/home/fteuser2/file.bin"/>
        </fte:postsrc>
    </fte:filecopy>

```

Related concepts

[“Timeout option for file transfers in recovery” on page 236](#)

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filemove Ant task

The **fte:filemove** task moves files between Managed File Transfer agents. When a file has been successfully transferred from the source agent to the destination agent, the file is deleted from the source agent.

Attributes

cmdqm

Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host*, *port*, or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see [“The MFT command.properties file” on page 522](#).

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see [com.ibm.wmqfte.propertySet](#).

If you do not use the `cmdqm` attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the `cmdqm` attribute, that is, *qmgrname@host@port@channel*.

dst

Required. Specifies the destination agent for the copy operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the destination agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

idproperty

Optional unless you have specified an outcome of `defer`. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted

and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.

You cannot specify this property if you have also specified an outcome property of `ignore`. However, you must specify `idproperty` if you have also specified an outcome property of `defer`.

jobname

Optional. Assigns a job name to the move request. You can use job names to create logical groups of transfers. Use the `fte:uuid` task to generate pseudo-unique job names. If you do not use the `jobname` attribute, the task defaults to using the `com.ibm.wmqfte.ant.jobName` property value, if this property is set. If you do not set this property, no job name is associated with the move request.

origuser

Optional. Specifies the originating user identifier to associate with the move request. If you do not use the `origuser` attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome

Optional. Determines whether the task waits for the move operation to complete before returning control to the Ant script. Specify one of the following options:

await

The task waits for the move operation to complete before returning. When an outcome of `await` is specified the `idproperty` attribute is optional.

defer

The task returns as soon as the move request has been submitted and assumes that the outcome of the move operation is dealt with later using either the [“fte:awaitoutcome Ant task” on page 822](#) or [“fte:ignoreoutcome Ant task” on page 833](#) task. When an outcome of `defer` is specified the `idproperty` attribute is required.

ignore

If the outcome of the move operation is not important, you can specify a value of `ignore`. The task then returns as soon as the move request has been submitted, without allocating any resources for tracking the outcome of the transfer. When an outcome of `ignore` is specified the `idproperty` attribute cannot be specified.

If you do not specify the outcome attribute, the task defaults to using the value `await`.

priority

Optional. Specifies the priority to associate with the move request. In general, higher priority transfer requests take precedence over lower priority requests. The priority value must be in the range 0 - 9 (inclusive). A priority value of 0 is the lowest priority and a value of 9 is the highest priority. If you do not specify the `priority` attribute, the transfer defaults to a priority of 0.

rcproperty

Optional. Specifies the name of a property to assign the result code of the move request to. The result code reflects the overall outcome of the move request.

You cannot specify this property if you have also specified an outcome property of `ignore` or `defer`. However, you must specify `rcproperty` if you have specified an outcome of `await`.

V 9.0.1 transferRecoveryTimeout

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

-1

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

0

The agent stops the file transfer as soon as it enters recovery.

>0

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src=agent1@qm1 dst="agent2@qm2"
  rcproperty="move.result" transferRecoveryTimeout="21600">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
  file.bin"/>
</fte:filemove
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. Maximum value for this attribute is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a property to the [The agent.properties file](#). For more information, see [“Timeout option for file transfers in recovery” on page 236](#).

src

Required. Specifies the source agent for the move operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the source agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

Parameters specified as nested elements

fte:filespec

Required. You must specify at least one file specification that identifies the files to move. You can specify more than one file specification if required. See the [fte:filespec](#) topic for more information.

fte:metadata

Optional. You can specify metadata to associate with the file move operation. This metadata is carried with the transfer and is recorded in the log messages generated by the transfer. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata. See the [fte:metadata](#) topic for more information.

fte:presrc

Optional. Specifies a program invocation to take place at the source agent before the transfer starts. You can only associate a single `fte:presrc` element with a given transfer. See the [program invocation](#) topic for more information.

fte:predst

Optional. Specifies a program invocation to take place at the destination agent before the transfer starts. You can only associate a single `fte:predst` element with a given transfer. See the [program invocation](#) topic for more information.

fte:postsrc

Optional. Specifies a program invocation to take place at the source agent after the transfer has completed. You can only associate a single `fte:postsrc` element with a given transfer. See the [program invocation](#) topic for more information.

fte:postdst

Optional. Specifies a program invocation to take place at the destination agent after the transfer has completed. You can only associate a single `fte:postdst` element with a given transfer. See the [program invocation](#) topic for more information.

If `fte:presrc`, `fte:predst`, `fte:postsrc`, `fte:postdst`, and `exits` do not return a success status, the rules are as follows in the order specified:

1. Run the source start exits. If source start exits fail the transfer fails and nothing further is run.
2. Run the pre-source call (when present). If the pre-source call fails, the transfer fails and nothing further is run.

3. Run the destination start exits. If the destination start exits fail the transfer fails and nothing further is run.
4. Run the pre-destination call (when present). If the pre-destination call fails, the transfer fails and nothing further is run.
5. Perform the file transfers.
6. Run the destination end exits. There is no failure status for these exits.
7. If the transfer is successful (if some files transfer successfully, the transfer is considered successful), run the post-destination call (if present). If the post-destination call fails, the transfer fails.
8. Run the source end exits. There is no failure status for these exits.
9. If the transfer is successful, run the post-source call (if present). If the post-source call fails, the transfer fails.

Examples

This example shows a basic file move between agent1 and agent2. The command to start the file move is sent to a queue manager called qm0, using a client transport mode connection. The result of the file transfer operation is assigned to the property called `move.result`.

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="move.result">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
</fte:filemove>
```

Related concepts

[“Timeout option for file transfers in recovery” on page 236](#)

You can set the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. If the transfer is not successful when the agent reaches the timeout for the retry interval, the transfer fails.

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:ignoreoutcome Ant task

Ignore the outcome of an **fte:filecopy**, **fte:filemove**, or **fte:call** command. When you specify an **fte:filecopy**, **fte:filemove**, or **fte:call** task to have an outcome of `defer`, the Ant task allocates resources to tracking this outcome. If you are no longer interested in the outcome, you can use the **fte:ignoreoutcome** task to free those resources.

Attributes

id

Required. Identifies the outcome that is no longer of interest. Typically you specify this identifier using a property that you set using the `idproperty` attribute of the [“fte:filecopy Ant task” on page 826](#), [“fte:filemove Ant task” on page 830](#), or [“fte:call Ant task” on page 823](#) task.

Example

This example shows how you can use the `fte:ignoreoutcome` task to free the resources allocated to tracking the outcome of the earlier [“fte:filecopy Ant task” on page 826](#) task.

```
<!-- issue a file copy request -->
```

```

<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent1@qm1"
              idproperty="copy.id"
              outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="{copy.id}"/>

```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:ping Ant task

This IBM MQ Managed File Transfer Ant task pings an agent to elicit a response, and so determines if the agent is able to process transfers.

Attributes

agent

Required. Specifies the agent to submit the **fte:ping** request to. The value is in the form: *agentname@qmgrname* where *agentname* is the name of the agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

cmdqm

Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host*, *port*, or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see [“The MFT command.properties file” on page 522](#).

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see [com.ibm.wmqfte.propertySet](#).

If you do not use the `cmdqm` attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the `cmdqm` attribute, that is, *qmgrname@host@port@channel*.

rcproperty

Required. Names a property to store the return code of the **ping** operation in.

timeout

Optional. The maximum amount of time, in seconds, for the task to wait for the agent to respond. The minimum timeout is zero seconds, however a timeout of minus one can also be specified such that the command waits forever for the agent to respond. If no value is specified for the `timeout` then the default is to wait up to 5 seconds for the agent to respond.

Example

This example sends a **fte:ping** request to agent1 hosted by qm1. The **fte:ping** request waits 15 seconds for the agent to respond. The outcome of the **fte:ping** request is stored in a property called ping.rc.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Return codes

0

Command completed successfully.

2

Command timed out.

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:uuid Ant task

Generates a pseudo-random unique identifier and assigns it to a given property. For example, you can use this identifier to generate job names for other file transfer operations.

Attributes

length

Required. The numeric length of UUID to generate. This length value does not include the length of any prefix, specified by the **prefix** parameter.

property

Required. The name of the property to assign the generated UUID to.

prefix

Optional. A prefix to add to the generated UUID. This prefix is not counted as part of the length of the UUID, as specified by the **length** parameter.

Example

This example defines a UUID that starts with the letters ABC followed by 16 pseudo-random hex characters. The UUID is assigned to a property named uuid.property.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filespec Ant nested element

The **fte:filespec** parameter is used as a nested element in other tasks. Use **fte:filespec** to describe a mapping between one or more source files, directories  **z/os** or data sets, and a

destination. Typically this element is used when expressing a set of files or directories  or data sets to move or copy.

Nested by:

- The [fte:filecopy](#) task
- The [fte:filemove](#) task

Source specification attributes

You must specify one of `srcfilespec` or `srcqueue`.

srcfilespec

Specifies the source of the file operation. The value of this attribute can include a wildcard.

srcqueue

Specifies the source of the transfer is a queue. The transfer moves data from messages stored on the queue specified by this attribute. You cannot specify this attribute if the **fte:filespec** task is nested within the **fte:filecopy** task.

The `srcqueue` attribute is not supported when the source agent is a protocol bridge agent.

Destination specification attributes

You must specify one of `dstdir`, `dstds`, `dstfilespace`, `dstfile`, `dstqueue` or `dstpds`.

dstdir

Specifies a directory as the destination for a file operation.

dstds

Specifies a data set as the destination for a file operation.

This attribute is supported only when the destination agent is running on the z/OS platform.

dstfile

Specifies a file as the destination for a file operation.

dstfilespace

Specifies a file space as the destination for a file operation.

This attribute applies only if the destination agent is a IBM MQ 8.0 web agent that has access to the web gateway file space.

dstpds

Specifies a partitioned data set as the destination for a file operation.

This attribute is supported only when the destination agent is running on the z/OS platform.

dstqueue

Specifies a queue as the destination for a file to message operation. You can optionally include a queue manager name in this specification, using the format `QUEUE@QUEUEMANAGER`. If you do not specify a queue manager name the destination agent queue manager is used if you have not set the `enableClusterQueueInputOutput` agent property to true. If the `enableClusterQueueInputOutput` property is set to true, the destination agent uses standard IBM MQ procedures to determine where the queue is located. You must specify a valid queue name that exists on the queue manager.

If you specify the `dstqueue` attribute, you cannot specify the `srcqueue` attributes because these attributes are mutually exclusive.

The `dstqueue` attribute is not supported when the destination agent is a protocol bridge agent.

Source option attributes

srcencoding

Optional. The character set encoding used by the file to transfer.

You can specify this attribute only when the conversion attribute is set to a value of `text`.

If you do not specify the `srcencoding` attribute, the character set of the source system is used for text transfers.

srceol

Optional. The end of line delimiter used by the file being transferred. The valid values are as follows:

- `CRLF` - Use a carriage return character followed by a line-feed character as the end of line delimiter. This convention is typical for Windows systems.
- `LF` - Use a line-feed character as the end of line delimiter. This convention is typical for UNIX systems.

You can specify this attribute only when the conversion attribute is set to a value of `text`. If you do not specify the `srceol` attribute, text transfers automatically determine the correct value based on the operating system of the source agent.

`srckeeptrailingspaces`

Optional. Determines whether trailing spaces are kept on source records read from a fixed-length-format data set as part of a text mode transfer. The valid values are as follows:

- `true` - trailing spaces are kept.
- `false` - trailing spaces are stripped.

If you do not specify the `srckeeptrailingspaces` attribute, a default value of `false` is specified.

You can specify this attribute only if you also specify the `srcfilespec` attribute and you set the conversion attribute to a value of `text`.

srcmsgdelimbytes

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by `x`. Multiple bytes must be comma-separated. For example, `srcmsgdelimbytes="x08,xA4"`. You can specify the `srcmsgdelimbytes` attribute only if you have also specified the `srcqueue` attribute. You cannot specify the `srcmsgdelimbytes` attribute if you have also specified the value `text` for the conversion attribute.

srcmsgdelimtext

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `srcmsgdelimtext="\u007d\n"`. The text delimiter is inserted after each message by the source agent. The text delimiter is encoded to binary format using the source encoding of the transfer. Each message is read in binary format, the encoded delimiter is appended in binary format to the message, and the result is transferred in binary format to the destination agent. If the source agent code page includes shift-in and shift-out states, the agent assumes that each message is in the shift-out state at the end of the message. At the destination agent the binary data is converted in the same way as a file to file text transfer. You can only specify the `srcmsgdelimtext` attribute if you have also specified the `srcqueue` attribute and a value of `text` for the conversion attribute.

srcmsgdelimposition

Optional. Specifies the position that the text or binary delimiter is inserted into. The valid values are as follows:

- `prefix` - the delimiters are inserted into the destination file before the data from each message.
- `postfix` - the delimiters are inserted into the destination file after the data from each message.

You can specify the `srcmsgdelimposition` attribute only if you have also specified one of the `srcmsgdelimbytes` or `srcmsgdelimtext` attributes.

srcmsggroups

Optional. Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this attribute is not specified, all messages on the source queue are written to the destination file. You can specify the `srcmsggroups` attribute only if you have also specified the `srcqueue` attribute.

srcqueuetimeout

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to be written to the queue.
- If the `srcmsggroups` attribute was specified, for a complete group to be written on the queue.

If neither of these conditions are met within the time specified by the value of `srcqueuetimeout`, the source agent stops reading from the queue and completes the transfer. If the `srcqueuetimeout` attribute is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the `srcmsggroups` attribute is specified, if there is no complete group on the queue. You can specify the `srcqueuetimeout` attribute only if you have also specified the `srcqueue` attribute.

For information about setting the `srcqueuetimeout` value, see [“Guidance for specifying a wait time on a message-to-file transfer”](#) on page 657.

z/OS

srcrecdelimbytes

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple records from a record-oriented source file to a binary file. You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. Multiple bytes must be comma-separated. For example:

```
srcrecdelimbytes="x08,xA4"
```

You can specify the `srcrecdelimbytes` attribute only if the transfer source file is record oriented, for example a z/OS data set, and the destination file is a normal, non-record-oriented file. You cannot specify the `srcrecdelimbytes` attribute if you have also specified the value `text` for the `conversion` attribute.

srcrecdelimpos

Optional. Specifies the position that the binary delimiter is inserted into. The valid values are as follows:

- `prefix` - the delimiters are inserted into the destination file before the data from each source record-oriented file record.
- `postfix` - the delimiters are inserted into the destination file after the data from each source record-oriented file record.

You can specify the `srcrecdelimpos` attribute only if you have also specified the `srcrecdelimbytes` attribute.

Destination option attributes

dstencoding

Optional. The character set encoding to use for the transferred file.

You can specify this attribute only when the `conversion` attribute is set to a value of `text`.

If the `dstencoding` attribute is not specified, the character set of the destination system is used for text transfers.

dsteol

Optional. The end of line delimiter to use for the transferred file. The valid values are as follows:

- **CRLF** - Use a carriage return character followed by a line-feed character as the end of line delimiter. This convention is typical for Windows systems.
- **LF** - Use a line-feed character as the end of line delimiter. This convention is typical for UNIX systems.

You can specify this attribute only when the conversion attribute is set to a value of `text`.

If you do not specify the `dsteol` attribute, text transfers automatically determine the correct value based on the operating system of the destination agent.

dstmsgdelimbytes

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a hexadecimal byte as a delimiter is `xNN`, where `N` is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: `x3e,x20,x20,xbf`.

You can specify the `dstmsgdelimbytes` attribute only if you have also specified the `dstqueue` attribute and the transfer is in binary mode. You can specify only one of the `dstmsgsize`, `dstmsgdelimbytes`, and `dstmsgdelimpattern` attributes.

dstmsgdelimpattern

Optional. Specifies the Java regular expression to use when splitting a text file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a regular expression as a delimiter is a regular expression enclosed in parentheses, (*regular_expression*), or enclosed in double quotation marks, "*regular_expression*". For more information, see [“Regular expressions used by MFT” on page 633](#).

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior using the `maxDelimiterMatchLength` agent property. For more information, see [“Advanced agent properties” on page 527](#).

You can specify the `dstmsgdelimpattern` attribute only if you have also specified the `dstqueue` attribute and the transfer is in text mode. You can specify only one of the `dstmsgsize`, `dstmsgdelimbytes`, and `dstmsgdelimpattern` attributes.

dstmsgdelimposition

Optional. Specifies the position that the text or binary delimiter is expected to be in. The valid values are as follows:

- `prefix` - The delimiters are expected at the beginning of each line.
- `postfix` - The delimiters are expected at the end of each line.

You can specify the `dstmsgdelimposition` attribute only if you have also specified the `dstmsgdelimpattern` attribute.

dstmsgincludedelim

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. If the `dstmsgincludedelim` attribute is specified, the delimiter is included at the end of the message that contains the file data preceding the delimiter. By default the delimiter is not included in the messages. You can specify the `dstmsgincludedelim` attribute only if you have also specified one of the `dstmsgdelimpattern` and `dstmsgdelimbytes` attributes.

dstmsgpersist

Optional. Specifies whether messages written to the destination queue are persistent. The valid values are as follows:

- `true` - Write persistent messages to the destination queue. This is the default value.

- `false` - Write non-persistent messages to the destination queue.
- `qdef` - The persistence value is taken from the `DefPersistence` attribute of the destination queue.

You can specify this attribute only when the `dstqueue` attribute is also specified.

dstmsgprops

Optional. Specifies whether the first message written to the destination queue by the transfer has IBM MQ message properties set. Possible values are:

- `true` - Set message properties on the first message created by the transfer.
- `false` - Do not set message properties on the first message created by the transfer. This is the default value.

For more information, see [“MQ message properties set by MFT on messages written to destination queues” on page 651.](#)

You can specify this attribute only when the `dstqueue` attribute is also specified.

dstmsgsize

Optional. Specifies whether to split the file into multiple fixed-length messages. All of the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The size of the messages is specified by the value of `dstmsgsize`. The format of `dstmsgsize` is *lengthunits*, where *length* is a positive integer value and *units* is one of the following values:

- `B` - Bytes. The minimum value allowed is two times the maximum bytes-per-character value of the code page of the destination messages.
- `K` - Kibibytes. This is equivalent to 1024 bytes.
- `M` - Mebibytes. This is equivalent to 1024 kibibytes.

If the file is transferred in text mode, and is in a double-byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can specify the `dstmsgsize` attribute only if you have also specified the `dstqueue` attribute. You can specify only one of the `dstmsgsize`, `dstmsgdelimbytes`, and `dstmsgdelimpattern` attributes.

dstunsupportedcodepage

Optional. Specifies the action to take if the destination queue manager, as specified by the `dstqueue` attribute, does not support the code page used when transferring file data to a queue as a text transfer. The valid values for this attribute are as follows:

- `binary` - continue the transfer but do not apply code page conversion to the data being transferred. Specifying this value is equivalent to not setting the conversion attribute to `text`.
- `fail` - do not continue with the transfer operation. The file is recorded as having failed to transfer. This is the default.

You can only specify the `dstunsupportedcodepage` attribute if you have also specified the `dstqueue` attribute and a value of `text` for the conversion attribute.

dsttruncaterecords

Optional. Specifies that destination records longer than the `LRECL` data set attribute are truncated. If set to `true`, the records are truncated. If set to `false`, the records are wrapped. The default setting is `false`. This parameter is valid only for text mode transfers where the destination is a data set.

Other attributes

checksum

Optional. Determines the algorithm used to checksum transferred files.

- `MD5` - use the MD5 hashing algorithm.
- `NONE` - do not use a checksum algorithm.

If you do not specify the checksum attribute, a default value of `MD5` is used.

conversion

Optional. Specifies the type of conversion to apply to the file as it is being transferred. Possible values are:

- `binary` - apply no conversion.
- `text` - apply code page conversion between the source and destination systems. Also apply conversion of line delimiters. The `srcencoding`, `dstencoding`, `srceol` and `dsteol` attributes influence the conversion that is applied.

If you do not specify the conversion attribute, a default value of `binary` is specified.

overwrite

Optional. Determines whether an existing destination file `> z/OS` or data set can be overwritten by the operation. When you specify a value of `true`, any existing destination file `> z/OS` or data sets are overwritten. When you specify a value of `false`, the existence of a duplicate file `> z/OS` or data set at the destination results in the operation failing. If the `overwrite` attribute is not specified, a default value of `false` is specified.

recurse

Optional. Determines whether the file transfer recurses into subdirectories. When you specify a value of `true`, the transfer recurses into subdirectories. When you specify a value of `false`, the transfer does not recurse into subdirectories. If the `recurse` attribute is not specified, a default value of `false` is specified.

Example

This example specifies a `fte:filespec` with a source file of `file1.bin` and a destination file of `file2.bin`.

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:metadata Ant nested element

Metadata is used to carry additional user-defined information with a file transfer operation.

See [“Metadata for MFT user exits” on page 845](#) for more information about how Managed File Transfer uses metadata.

Nested by:

- The [fte:filecopy](#) task
- The [fte:filemove](#) task
- The [fte:call](#) task

Parameters specified as nested elements

fte:entry

You must specify at least one entry inside the `fte:metadata` nested element. You can choose to specify more than one entry. Entries associate a key name with a value. Keys must be unique in a block of `fte:metadata`

Entry attributes

name

Required. The name of the key belonging to this entry. This name must be unique across all **entry** parameters nested inside a `fte:metadata` element.

value

Required. The value to assign to this entry.

Example

This example shows a `fte:metadata` definition that contains two entries.

```
<fte:metadata>
  <fte:entry name="org.foo.partColor" value="red"/>
  <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Program invocation nested elements

Programs can be started using one of five nested elements: `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc`, and `fte:command`. These nested elements instruct an agent to call an external program as part of its processing. Before you can start a program, you must ensure that the command is in a location specified by the `commandPath` property in the `agent.properties` file of the agent that runs the command.

Even though each program invocation element has a different name, they share the same set of attributes and the same set of nested elements. Programs can be started by the **`fte:filecopy`**, **`fte:filemove`**, and **`fte:command`** Ant tasks.

You cannot invoke programs from a Connect:Direct bridge agent.

Ant tasks that can invoke programs:

- The `fte:filecopy` task nests program invocation parameters using the `fte:predst`, `fte:postdst`, `fte:presrc`, and `fte:postsrc` nested elements.
- The `fte:filemove` task nests program invocation parameters using the `fte:predst`, `fte:postdst`, `fte:presrc`, and `fte:postsrc` nested elements.
- The `fte:call` task nests program invocation parameters using the `fte:command` nested element.

Attributes

command

Required. Names the program to call. For the agent to be able to run a command, the command must be in a location specified by the `commandPath` property in the agent's `agent.properties` file. For more information, see [“commandPath MFT property” on page 347](#). Any path information specified in the `command` attribute is considered relative to a location specified by the `commandPath` property. When `type` is `executable`, an executable program is expected otherwise a script appropriate for the `call` type is expected.

retrycount

Optional. The number of times to retry calling the program if the program does not return a success return code. The program named by the `command` attribute is called up to this number of times.

The value assigned to this attribute must be non-negative. If you do not specify the `retrycount` attribute, a default value of zero is used.

retrywait

Optional. The time to wait, in seconds, before trying the program invocation again. If the program named by the `command` attribute does not return a success return code and the `retrycount` attribute specifies a non-zero value, this parameter determines the time to wait between retries. The value assigned to this attribute must be non-negative. If you do not specify the `retrywait` attribute, a default value of zero is used.

successrc

Optional. The value of this attribute is used to determine when the program invocation successfully runs. The process return code for the command is evaluated using this expression. The value can be composed of one or more expressions combined with a vertical bar character (|) to signify Boolean OR, or an ampersand (&) character to signify Boolean AND. Each expression can be one of the following types of expression:

- A number to indicate an equality test between the process return code and the number.
- A number prefixed with a ">" character to indicate a greater-than test between the number and the process return code.
- A number prefixed with a "<" character to indicate a less-than test between the number and the process return code.
- A number prefixed with a "!=" character to indicate a not-equal-to test between the number and the process return code.

For example: `>2&<7&!5|0|14` is interpreted as the following return codes being successful: 0, 3, 4, 6, 14. All other return codes are interpreted as being unsuccessful. If you do not specify the `successrc` attribute, a default value of zero is used. This means that the command is judged to have successfully run if, and only if, it returns a code of zero.

type

Optional. The value of this attribute specifies what type of program is being called. Specify one of the following options:

executable

The task calls an executable program. Can have additional arguments specified using the `arg` nested element. The program is expected to be accessible on the `commandPath` and where applicable have execute permission set. UNIX scripts can be called as long as they specify a shell program (for example, first line of shell script file is: `#!/bin/sh`). Command output written to `stderr` or `stdout` is sent to the Managed File Transfer log for the call. However, the amount of data output is limited by the agent configuration. The default is 10K bytes of data, but you can override this default using the agent property: `maxCommandOutput`.

antscript

The task runs the specified Ant script, using the `fteAnt` command. Properties can be specified using the property nested element. Ant targets can be specified using the `target` nested element. The Ant script is expected to be accessible on the `commandPath`. Ant output written to `stderr` or `stdout` is sent to the Managed File Transfer log for the call. However, the amount of data output is limited by the agent configuration. The default is 10K bytes of data but you can override this default using the agent property: `maxCommandOutput`.

jcl

The value `jcl` is supported on z/OS only and runs the specified z/OS JCL script. The JCL is submitted as a job and requires the job card to be present. When the job is submitted successfully the JCL command output, written to the Managed File Transfer log, contains the following text: `JOB job_name(job_id)` where:

- `job_name` is the name of the job identified by the job card in the JCL.
- `job_id` is the z/OS system generated job ID.

If the job cannot be submitted successfully, the JCL script command fails and writes a message to the log indicating the reason for the failure (for example no job card is present). To understand whether the job has been run or completed successfully, use a system service such as SDSF. Managed File Transfer does not provide the information because it only submits the job; the system then determines when to run the job and how the job output is presented. Because a JCL script is submitted as a batch job it is not advisable to specify `jcl` for a `presrc` or `predst` nested element because you only know that the job has been submitted successfully and not whether it ran to completion successfully before the transfer starts. There are no nested elements that are valid with a type of `jcl`.

The following example shows a JCL job:

```
//MYJOB JOB
//*
//MYJOB EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=H
//SYSUT1 DD DSN=FRED.DEMO.TXT,DISP=SHR
//SYSUT2 DD DSN=BOB.DEMO.TXT,DISP=(NEW,CATLG),
// RECFM=VB,LRECL=133,BLKSIZE=2048,
// SPACE=(TRK,(30,5),RLSE)
//SYSIN DD DUMMY
```

Parameters specified as nested elements

fte:arg

Only valid where the value of the `type` attribute is `executable`. Use nested `fte:arg` elements to specify arguments to the program that is being called as part of the program invocation. The program arguments are built from the values specified by the `fte:arg` elements in the order that the `fte:arg` elements are encountered. You can choose to specify zero or more `fte:arg` elements as nested elements of a program invocation.

fte:property

Only valid where the value of the `type` attribute is `antscript`. Use the `name` and `value` attributes of the nested `fte:property` elements to pass in name-value pairs to the Ant script. You can choose to specify zero or more `fte:property` elements as nested elements of a program invocation.

fte:target

Only valid where the value of the `type` attribute is `antscript`. Specify a target in the Ant script to call. You can choose to specify zero or more `fte:target` elements as nested elements of a program invocation.

Arg attributes

value

Required. The value of the argument to pass to the program being called.

Property attributes

name

Required. The name of a property to pass to the Ant script.

value

Required. The value to associate with the property name being passed to the Ant script.

Examples

This example shows an `fte:postsrc` program invocation being specified as part of an `fte:filecopy` task. The program invocation is for a program called `post.sh` and is supplied a single argument of `/home/fteuser2/file.bin`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
rcproperty="copy.result">
```

```

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
    <fte:postsrc command="post.sh" successsrc="1" >
        <fte:arg value="/home/fteuser2/file.bin"/>
    </fte:postsrc>
</fte:filecopy>

```

This example shows an `fte:command` program invocation being specified as part of a `fte:call` task. The program invocation is for an executable called `command.sh`, which is not passed any command-line arguments. If `command.sh` does not return a success return code of 1, the command is tried again after 30 seconds.

```

<fte:call cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
    agent="agent1@qm1"
    rcproperty="call.rc"
    origuser="bob"
    jobname="{j}job.id">
    <fte:command command="command.sh" successsrc="1" retrycount="5" retrywait="30"/>
</fte:call>

```

This example shows an `fte:command` program invocation being specified as part of a `fte:call` task. The program invocation is for the copy and compress targets in an Ant script called `script.xml`, which is passed two properties.

```

<fte:call cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
    agent="agent1@qm1"
    rcproperty="call.rc"
    origuser="bob"
    jobname="{j}job.id">
    <fte:command command="script.xml" type="antscript">
        <property name="src" value="AGENT5@QM5"/>
        <property name="dst" value="AGENT3@QM3"/>
        <target name="copy"/>
        <target name="compress"/>
    </fte:command>
</fte:call>

```

Related concepts

[“Specifying programs to run with MFT” on page 239](#)

You can run programs on a system where a Managed File Transfer Agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference

[“Using Apache Ant with MFT” on page 241](#)

Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

[“Ant tasks provided by MFT” on page 822](#)

Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Working with MFT user exits for customization

Metadata for MFT user exits

There are three different types of metadata that can be supplied to user exit routines for Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

Environment metadata

Environment metadata is passed to all user exit routines and describes the agent runtime environment that the user exit routine is being called from. This metadata is read-only and cannot be updated by any user exit routine.

<i>Table 71. Environment metadata</i>	
Key	Description
AGENT_CONFIGURATION_DIRECTORY_KEY	The name of the directory that contains the agent's configuration information.
AGENT_PRODUCT_DIRECTORY_KEY	The name of the directory that the agent code has been installed in.
AGENT_VERSION_KEY	Version number for the agent runtime that calls the exit routine.

The key names and value names given in Table 1 are constants that are defined in the EnvironmentMetaDataConstants interface.

Transfer metadata

Transfer metadata is passed to all user exit routines. The metadata consists of system-supplied values and user-supplied values. If you change any system-supplied values, these changes are ignored. The initial user-supplied values for the source transfer start user exit are based on those values you supply when you define the transfer. The source agent can change user-supplied values as part of the processing of the source transfer start user exit. This user exit is called before the entire file transfer starts. These changes are used in subsequent calls to other exit routines that relate to that transfer. Transfer metadata is applied to an entire transfer.

Although all user exits can read values from the transfer metadata, only the source transfer start user exit can change transfer metadata

You cannot use transfer metadata to propagate information between different file transfers.

The system-supplied transfer metadata is detailed in Table 2:

<i>Table 72. Transfer metadata</i>	
Key	Description
DESTINATION_AGENT_KEY	The name of the agent that is the destination for the transfer.
JOB_NAME_KEY	The job name associated with the transfer request
MQMD_USER_KEY	The MQMD user field from the message used to submit the transfer request
ORIGINATING_HOST_KEY	The host name specified as the originating host name in the transfer request
ORIGINATING_USER_KEY	The user name specified as the originating user ID in the transfer request
SOURCE_AGENT_KEY	The name of the agent that is the source of the transfer
TRANSFER_ID_KEY	The identifier of the transfer

The key names and value names given in Table 2 are constants that are defined in the TransferMetaDataConstants interface.

File metadata

The file metadata is passed to the source transfer start exit as part of the file specification. There is separate file metadata for the source and destination files.

You cannot use file metadata to propagate information between different file transfers.

Table 73. File metadata

Key	Permitted values	Description
CONVERT_LINE_SEPARATORS		Key value used for text transfers to indicate whether CRLF (carriage return-line feed) or LF (line feed) line separator sequences in source data are converted to the line separator sequence at the destination.
DELIMITER_KEY		Key value used to define a delimiter to separate record data when transferring record-oriented data to normal files. Also used for message-to-file and file-to-message transfers.
DELIMITER_POSITION_KEY	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	Use with the DELIMITER_KEY to define the position of the delimiter; either prefix or postfix.
DELIMITER_TYPE_KEY	DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE DELIMITER_TYPE_SIZE_VALUE	Use with the DELIMITER_KEY to define the type of delimiter.
DESTINATION_EXIST_KEY	DESTINATION_EXIST_KEY_ERROR_VALUE DESTINATION_EXIST_KEY_OVERWRITE_VALUE	Determines the file transfer behavior if the destination file exists.
FILE_ALIAS_KEY		Key value used to define an alias for the file being transferred.
FILE_CHECKSUM_METHOD_KEY	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Determines the checksum method to use when transferring the file.
FILE_CONVERSION_KEY	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	Determines the type of conversion applied to the file contents.
FILE_ENCODING_KEY		Determines the encoding used for a text file.
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	Determines the character sequence that denotes the end of a line: <LF> or <CR><LF>.
FILE_SPACE_ALIAS		Determines the alias of a file in the file space. Note: This metadata can be used only if the FILE_TYPE_KEY is FILE_TYPE_FILE_SPACE_VALUE
FILE_SPACE_NAME		Determines the name of the file space. Note: This metadata can be used only if the FILE_TYPE_KEY is FILE_TYPE_FILE_SPACE_VALUE
FILE_TYPE_KEY	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Determines the destination file, queue, or file space specification.
GROUP_ID_KEY		Key value used for message-to-file transfers to determine the group of messages to read from the source queue. This attribute is valid only when the value of USE_GROUPS_KEY is USE_GROUPS_TRUE_VALUE.

Table 73. File metadata (continued)		
Key	Permitted values	Description
INCLUDE_DELIMITER_IN_MESSAGE_KEY	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	Key value used for file-to-message transfers to determine whether to include the delimiters that were used to split the file into multiple messages at the end of the messages. This attribute is valid only when the value of DELIMITER_TYPE_KEY is DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE.
INSERT_RECORD_LINE_SEPARATOR_KEY		Key value used for text transfers from record-oriented files to specify whether line separators are inserted into the data after each record.
KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	Key value used to determine whether trailing spaces are removed from records read from fixed-length-format data sets.
NEW_RECORD_ON_LINE_SEPARATOR_KEY		Key value used for text transfers to record-oriented files to specify whether line separators in the data are included with the record data or cause a new record (and are not written).
PERSISTENT_KEY	PERSISTENT_TRUE_VALUE PERSISTENT_FALSE_VALUE PERSISTENT_QDEF_VALUE	Key value used for file-to-message transfers to determine whether the messages are persistent.
SET_MQ_PROPS_KEY	SET_MQ_PROPS_TRUE_VALUE SET_MQ_PROPS_FALSE_VALUE	Key value used for file-to-message transfers to determine whether IBM MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
UNRECOGNISED_CODE_PAGE_KEY	UNRECOGNISED_CODE_PAGE_FAIL_VALUE UNRECOGNISED_CODE_PAGE_BINARY_VALUE	Key value used for file-to-message transfers to determine whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager.
USE_GROUPS_KEY	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	Key value used for message-to-file transfers to determine whether to transfer only a complete group of messages from the source queue.
WAIT_TIME_KEY		Key value used for message-to-file transfers to determine the time, in seconds, for the source agent to wait for one of the following cases: <ul style="list-style-type: none"> A message to appear on the source queue, if the queue is empty or has become empty, if the value of USE_GROUPS_KEY is FALSE. A complete group to appear on the source queue, if the value of USE_GROUPS_KEY is TRUE.

The key names and value names given in Table 3 are constants that are defined in the FileMetaDataConstants interface.

MFT resource monitor user exits

Resource monitor user exits allow you to configure custom code to run when a monitor's trigger condition is satisfied, before the associated task is started.

It is not recommended to invoke new transfers directly from user exit code. In some circumstances this causes files to be transferred multiple times as user exits are not resilient to agent restarts.

Resource monitor user exits use the existing infrastructure for user exits. The monitor user exits are called after a monitor has triggered but before the corresponding task has been run by the monitor's task. This allows the user exit to modify the task to be run and decide whether a task should proceed or not. You can modify the monitor task by updating the monitor metadata, which is then used for variable substitution in the task document created by the creation of the original monitor. Alternatively, the monitor exit can replace or update the task definition XML string passed as a parameter. The monitor exit can return a result code of either 'proceed' or 'cancel' for the task. If cancel is returned, the task will not be started and the monitor will not start again until the monitored resource matches the trigger conditions. If the resource has not changed, the trigger will not start. As with the other user exits, you can chain monitor exits together. If one of the exits returns a cancel result code, the overall result is cancel and the task is not started.

- A map of environment metadata (same as other user exits)
- A map of monitor metadata including immutable system metadata and mutable user metadata. The immutable system metadata is as follows:
 - FILENAME - name of the file that satisfied the trigger condition
 - FILEPATH - path to the file that satisfied the trigger condition
 - FILESIZE (in bytes - this metadata might not be present) - size of the file that satisfied the trigger condition
 - LASTMODIFIEDDATE (Local) - date that the file that satisfied the trigger condition was last changed. This date is expressed as the local date of the time zone the agent is running in and is formatted as an ISO 8601 date.
 - LASTMODIFIEDTIME (Local) - time in local format that the file that satisfied the trigger condition was last changed. This time is expressed as the local time of the time zone the agent is running in and is formatted as an ISO 8601 time.
 - LASTMODIFIEDDATEUTC - date in universal format that the file that satisfied the trigger condition was last changed. This date is expressed as the local date converted to the UTC time zone and is formatted as an ISO 8601 date.
 - LASTMODIFIEDTIMEUTC - time in universal format that the file that satisfied the trigger condition was last changed. This time is expressed the local time converted to the UTC time zone and is formatted as an ISO 8601 time.
 - AGENTNAME - the monitor agent name
- An XML string representing the task to be run as a result of the monitor trigger.

Monitor exits return the following data:

- An indicator that specifies whether to progress further (proceed or cancel)
- A string to insert into the trigger-satisfied log message

As a result of running the monitor exit code, the monitor metadata and task definition XML string that were originally passed as parameters might also have been updated.

The value of the agent property `monitorExitClasses` (in the agent `.properties` file) specifies which monitor exit classes to load, with each exit class separated by a comma. For example:

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

The interface to the monitor user exit is:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constant
     *      defined in <code>EnvironmentMetaDataConstants</code> class can
     *      be used to access the data held by this map.
     *
     * @param monitorMetaData
     *      meta data to associate with the monitor. The meta data passed
     *      to this method can be altered, and the changes will be
     *      reflected in subsequent exit routine invocations. This map
     *      also contains keys with IBM reserved names. These entries are
     *      defined in the <code>MonitorMetaDataConstants</code> class and
     *      have special semantics. The the values of the IBM reserved names
     *      cannot be modified by the exit
     *
     * @param taskDetails
     *      An XML String representing the task to be executed as a result of
     *      the monitor triggering. This XML string may be modified by the
     *      exit
     *
     * @return
     *      a monitor exit result object which is used to determine if the
     *      task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                               Map<String, String> monitorMetaData,
                               Reference<String> taskDetails);
}

```

The constants for the IBM-reserved values in the monitor metadata are as follows:

```
package com.ibm.wmqfte.exitroutine.api;

/**
 * Constants for IBM reserved values placed into the monitor meta data
 * maps used by the monitor exit routines.
 */
public interface MonitorMetaDataConstants {

    /**
     * The value associated with this key is the name of the trigger
     * file associated with the monitor. Any modification performed
     * to this property by user exit routines will be ignored.
     */
    final String FILE_NAME_KEY = "FILENAME";

    /**
     * The value associated with this key is the path to the trigger
     * file associated with the monitor. Any modification performed
     * to this property by user exit routines will be ignored.
     */
    final String FILE_PATH_KEY = "FILEPATH";

    /**
     * The value associated with this key is the size of the trigger
     * file associated with the monitor. This will not be present in
     * the cases where the size cannot be determined. Any modification
     */
}

```

```

* performed to this property by user exit routines will be ignored.
*/
final String FILE_SIZE_KEY = "FILESIZE";

/**
 * The value associated with this key is the local date on which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
 * The value associated with this key is the local time at which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";

/**
 * The value associated with this key is the UTC date on which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
 * The value associated with this key is the UTC time at which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
 * The value associated with this key is the name of the agent on which
 * the monitor is running. Any modification performed to this property by
 * user exit routines will be ignored.
 */
final String MONITOR_AGENT_KEY = "AGENTNAME";
}

```

Example monitor user exit

This example class implements the MonitorExit interface. This example adds a custom substitution variable into the monitor metadata called *REDIRECTEDAGENT* that will be populated with a value of LONDON if the hour of the day is odd, and a value of PARIS for even hours. The monitor exit result code is set to always return proceed.

```

package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**
 * Example resource monitor user exit that changes the monitor mutable
 * metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
 */
public class TestMonitorExit implements MonitorExit {

    // custom variable that will substitute destination agent
    final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

    public MonitorExitResult onMonitor(
        Map<String, String> environmentMetaData,
        Map<String, String> monitorMetaData,
        Reference<String> taskDetails) {

        // always succeed

```

```

        final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;

        final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

        if (hour%2 == 1) {
            monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
        } else {
            monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
        }

        return result;
    }
}

```

The corresponding task for a monitor that makes use of the *REDIRECTEDAGENT* substitution variable could look similar to the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
      QMgr="QM1"/>
    <destinationAgent agent="{REDIRECTEDAGENT}"
      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>

```

Before this transfer is started, the value of the `<destinationAgent>` element's `agent` attribute is replaced with either LONDON or PARIS.

You must specify the substitution variable in the monitor exit class and the task definition XML in uppercase.

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

[“Metadata for MFT user exits” on page 845](#)

There are three different types of metadata that can be supplied to user exit routines for Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

[“Java interfaces for MFT user exits” on page 855](#)

Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference

[“MFT Agent properties for user exits” on page 853](#)

In addition to the standard properties in the `agent.properties` file, there are several advanced properties specifically for user exit routines. These properties are not included by default so if you want to use any of them, you must manually edit the `agent.properties` file. If you make a change to `agent.properties` file while that agent is running, stop and restart the agent to pick up the changes.

MFT Agent properties for user exits

In addition to the standard properties in the `agent.properties` file, there are several advanced properties specifically for user exit routines. These properties are not included by default so if you want to use any of them, you must manually edit the `agent.properties` file. If you make a change to `agent.properties` file while that agent is running, stop and restart the agent to pick up the changes.

For IBM WebSphere MQ 7.5 or later, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see [“The use of environment variables in MFT properties”](#) on page 511.

User exit routine properties

The user exit routines are called in the order listed in the following table. For more information about the `agent.properties` file, see [Advanced agent properties: User exit routine](#).

Property name	Description
<code>sourceTransferEndExitClasses</code>	Specifies a comma-separated list of classes that implement a source transfer end exit routine.
<code>sourceTransferStartExitClasses</code>	Specifies a comma-separated list of classes that implement a source transfer start exit routine.
<code>destinationTransferStartExitClasses</code>	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.
<code>destinationTransferEndExitClasses</code>	Specifies a comma-separated list of classes that implement a destination transfer user exit routine.
<code>exitClassPath</code>	<p>Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines.</p> <p>The agent exits directory is searched before any entries in this class path.</p> <p>If you are using this property on Windows, use a forward slash character (/) as a path delimiter, not the backslash character (\). For example:</p> <pre>exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.</pre> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>
<code>exitNativeLibraryPath</code>	<p>Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.</p> <p>For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.</p>
<code>monitorExitClasses</code>	Specifies a comma-separated list of classes that implement a monitor exit routine. For more information, see “MFT resource monitor user exits” on page 849.
<code>protocolBridgeCredentialExitClasses</code>	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see “Mapping credentials for a file server by using exit classes” on page 214.
<code>protocolBridgePropertiesExitClasses</code>	Specifies a comma-separated list of classes that implement a protocol bridge server properties user exit routine. For more information, see “ProtocolBridgePropertiesExit2: Looking up protocol file server properties” on page 209.
<code>IOExitClasses</code>	Specifies a comma-separated list of classes that implement an I/O user exit routine. List only the classes that implement the <code>IOExit</code> interface, that is, do not list classes that implement the other I/O user exit interfaces, for example <code>IOExitResourcePath</code> and <code>IOExitChannel</code> . For more information, see “Using MFT transfer I/O user exits” on page 249.

Order of exit invocation

The source and destination exits are invoked in the following order:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Chaining source and destination exits

If you specify multiple exits, the first exit in the list is invoked first, followed by the second exit, and so on. Any changes made by the first exit are passed as input to the exit that is subsequently invoked and so on. For example, if there are two source transfer start exits any changes made to the transfer metadata by the first exit are input to the second exit. Each exit returns its own result. If all the exits of a given type return PROCEED as a transfer result code, the overall result is PROCEED. If one or more exits return CANCEL_TRANSFER, the overall result is CANCEL_TRANSFER. All of the result codes and strings returned by the exits are output in the transfer log.

If the overall result from the source transfer start exit is PROCEED, the transfer proceeds using any changes made by the exits. If the overall result is CANCEL_TRANSFER, the source transfer end exits are invoked and then the transfer is canceled. The completion status in the transfer log is "cancelled".

If the overall result from the destination transfer start exits is PROCEED, the transfer proceeds using any changes made by the exits. If the overall result is CANCEL_TRANSFER, the destination transfer end exits are invoked, then the source transfer end exits are invoked. Finally the transfer is canceled. The completion status in the transfer log is "cancelled".

If a source or destination exit needs to pass information to following exits either in the chain or in the order of execution it must be done by updating the transfer metadata. The usage of the transfer metadata is exit implementation specific. For instance, if an exit sets the return result to CANCEL_TRANSFER and needs to communicate to the following exits that the transfer has been canceled it must be done by setting a transfer metadata value in a way understood by the other exits.

Example

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar;C:/mycomp/mqft/exits/fileFilter.jar
```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

[“Metadata for MFT user exits” on page 845](#)

There are three different types of metadata that can be supplied to user exit routines for Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

[“Java interfaces for MFT user exits” on page 855](#)

Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference

[“MFT resource monitor user exits” on page 849](#)

Resource monitor user exits allow you to configure custom code to run when a monitor's trigger condition is satisfied, before the associated task is started.

[“The use of environment variables in MFT properties” on page 511](#)

From IBM WebSphere MQ 7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

[“The MFT agent.properties file” on page 526](#)

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Java interfaces for MFT user exits

Use the topics in this section for reference information about Java interfaces for user exit routines.

CDCredentialExit.java interface

CDCredentialExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing. This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the IBM MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

    /**
     * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *      The values of properties defined for the Connect:Direct bridge.
     *      These values can only be read, they cannot be updated by
     *      the implementation.
     *
     * @return
     *      true if the initialisation is successful and false if unsuccessful
     *      If false is returned from an exit the Connect:Direct bridge agent does not
     *      start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked once per transfer to map the IBM MQ user ID in the transfer message to the
     * credentials to be used to access the Connect:Direct node.
     *
     * @param mqUserId
     *      The IBM MQ user ID from which to map to the credentials to be used
     *      to access the Connect:Direct node
     * @param snode
     *      The name of the Connect:Direct SNODE specified as the cdNode in the
     *      file path. This is used to map the correct user ID and password for the
     *      SNODE.
     *
     * @return
     *      A credential exit result object that contains the result of the map and
     *      the credentials to use to access the Connect:Direct node
     */
    public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);
}
```

```

/**
 * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
 * any resources that were allocated by the exit
 *
 * @param bridgeProperties
 *       The values of properties defined for the Connect:Direct bridge.
 *       These values can only be read, they cannot be updated by
 *       the implementation.
 *
 * @return
 */
public void shutdown(final Map<String, String> bridgeProperties);    }

```

CredentialExitResult.java interface

CredentialExitResult.java

```

/**
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a Credential mapMQUserId exit method. It is composed of a result
 * code, which determines whether the mapping of the user id was successful, and an optional
 * Credentials object if the mapping is successful.
 */
public class CredentialExitResult {

    private final CredentialExitResultCode resultCode;
    private final Credentials credentials;

    /**
     * Constructor. Creates a credential exit result object with a specified result
     * code and optionally credentials.
     *
     * @param resultCode
     *       The result code to associate with the exit result being created.
     *
     * @param credentials
     *       The credentials to associate with the exit result being created.
     *       A value of <code>null</code> can be specified to indicate no
     *       credentials. If the resultCode is USER_SUCCESSFULLY_MAPPED the
     *       credentials must be set to a non-null value,
     */
    public CredentialExitResult(CredentialExitResultCode resultCode, Credentials credentials) {
        this.resultCode = resultCode;
        this.credentials = credentials;
    }

    /**
     * Returns the result code associated with this credential exit result
     *
     * @return    the result code associated with this exit result.
     */
    public CredentialExitResultCode getResultCode() {
        return resultCode;
    }

    /**
     * Returns the credentials associated with this credential exit result
     *
     * @return    the explanation associated with this credential exit result.
     */
    public Credentials getCredentials() {
        return credentials;
    }
}

```

```
}  
}
```

Related tasks

[Customizing MFT with user exits](#)

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

DestinationTransferEndExit.java interface

DestinationTransferEndExit.java

```
/*  
 * Licensed Materials - Property of IBM  
 *  
 * "Restricted Materials of IBM"  
 *  
 * 5724-H72  
 *  
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.  
 *  
 * US Government Users Restricted Rights - Use, duplication or  
 * disclosure restricted by GSA ADP Schedule Contract with  
 * IBM Corp.  
 */  
package com.ibm.wmqfte.exitpoint.api;  
  
/**  
 * An interface that is implemented by classes that want to be invoked as part of  
 * user exit routine processing. This interface defines a method that will be  
 * invoked immediately after completing a transfer on the agent acting as the  
 * destination of the transfer.  
 */  
public interface DestinationTransferEndExit {  
  
    /**  
     * Invoked immediately after the completion of a transfer on the agent acting as  
     * the destination of the transfer.  
     *  
     * @param transferExitResult  
     *        a result object reflecting whether or not the transfer completed  
     *        successfully.  
     *  
     * @param sourceAgentName  
     *        the name of the agent acting as the source of the transfer.  
     *  
     * @param destinationAgentName  
     *        the name of the agent acting as the destination of the  
     *        transfer. This is the name of the agent that the  
     *        implementation of this method will be invoked from.  
     *  
     * @param environmentMetaData  
     *        meta data about the environment in which the implementation  
     *        of this method is running. This information can only be read,  
     *        it cannot be updated by the implementation. The constants  
     *        defined in EnvironmentMetaDataConstants class can  
     *        be used to access the data held by this map.  
     *  
     * @param transferMetaData  
     *        meta data to associate with the transfer. The information can  
     *        only be read, it cannot be updated by the implementation. This  
     *        map may also contain keys with IBM reserved names. These  
     *        entries are defined in the TransferMetaDataConstants  
     *        class and have special semantics.  
     *  
     * @param fileResults  
     *        a list of file transfer result objects that describe the source  
     *        file name, destination file name and result of each file transfer
```

```

*           operation attempted.
*
* @return   an optional description to enter into the log message describing
*           transfer completion. A value of <code>null</code> can be used
*           when no description is required.
*/
String onDestinationTransferEnd(TransferExitResult transferExitResult,
                                String sourceAgentName,
                                String destinationAgentName,
                                Map<String, String>environmentMetaData,
                                Map<String, String>transferMetaData,
                                List<FileTransferResult>fileResults);
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“SourceTransferEndExit.java interface” on page 883](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

DestinationTransferStartExit.java interface

DestinationTransferStartExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param sourceAgentName
     *           the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *           the name of the agent acting as the destination of the
     *           transfer. This is the name of the agent that the
     *           implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *           meta data about the environment in which the implementation
     *           of this method is running. This information can only be read,
     *           it cannot be updated by the implementation. The constants
     *           defined in <code>EnvironmentMetaDataConstants</code> class can
     *           be used to access the data held by this map.
     */
}

```

```

* @param transferMetaData
*     meta data to associate with the transfer. The information can
*     only be read, it cannot be updated by the implementation. This
*     map may also contain keys with IBM reserved names. These
*     entries are defined in the <code>TransferMetaDataConstants</code>
*     class and have special semantics.
*
* @param fileSpecs
*     a list of file specifications that govern the file data to
*     transfer. The implementation of this method can modify the
*     entries in this list and the changes will be reflected in the
*     files transferred. However, new entries may not be added and
*     existing entries may not be removed.
*
* @return     a transfer exit result object which is used to determine if the
*             transfer should proceed, or be cancelled.
*/
TransferExitResult onDestinationTransferStart(String sourceAgentName,
                                             String destinationAgentName,
                                             Map<String, String> environmentMetaData,
                                             Map<String, String> transferMetaData,
                                             List<Reference<String>> fileSpecs);

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“SourceTransferEndExit.java interface” on page 883](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

FileTransferResult.java interface

FileTransferResult.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * Result information about a file transfer.
 */
public interface FileTransferResult {

    /** An enumeration for the <code>getCorrelatorType()</code> method. */
    public enum CorrelationInformationType {
        /** No correlation information is available for this result */
        NONE,
        /**
         * The correlation information relates to work done in
         * IBM Sterling File Gateway.
         */
        SFG
    }
}

/**

```

```

    * Returns the source file specification, from which the file was transferred.
    *
    * @return    the source file specification, from which the file was
    *            transferred.
    */
String getSourceFileSpecification();

/**
 * Returns the destination file specification, to which the file was transferred.
 *
 * @return    the destination file specification, to which the file was
 *            transferred. A value of <code>null</code> may be returned
 *            if the transfer did not complete successfully.
 */
String getDestinationFileSpecification();

/**
 * Returns the result of the file transfer operation.
 *
 * @return    the result of the file transfer operation.
 */
FileExitResult getExitResult();

/**
 * @return an enumerated value that identifies the product to which this correlating
 *         information relates.
 */
CorrelationInformationType getCorrelatorType();

/**
 * @return the first string component of the correlating identifier that relates
 *         this transfer result to work done in another product. A value of null
 *         may be returned either because the other product does not utilize a
 *         string based correlation information or because there is no correlation
 *         information.
 */
String getString1Correlator();

/**
 * @return the first long component of the correlating identifier that relates
 *         this transfer result to work done in another product. A value of zero
 *         is returned when there is no correlation information or the other
 *         product does not utilize long based correlation information or because
 *         the value really is zero!
 */
long getLong1Correlator();
}

```

Related tasks

[Customizing MFT with user exits](#)

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

IOExit.java interface

IOExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with

```

```

*   IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a
 * {@link #newPath(String)} method invocation will then be used by the WMQFTE
 * transfer to obtain information about the resource and to transfer data to or
 * from the resource.
 * <p>
 * To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
 * or {@link DestinationTransferStartExit} as appropriate, should be installed
 * to enable information to be seen by this exit. The
 * {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
 * passed the transfer's environment, metadata, and a list of file
 * specifications for the transfer. The paths for the file specifications are
 * the paths passed to the I/O exit's {@link #newPath(String)} method.
 * <p>
 * Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
 * methods might be called at other times by a WMQFTE agent and not just during
 * transfers. For example, at transfer setup time the I/O system is queried to
 * resolve the full resource paths for transfer.
 */
public interface IOExit {

    /**
     * Invoked once when the I/O exit is first required for use. It is intended
     * to initialize any resources that are required by the exit.
     *
     * @param agentProperties
     *     The values of properties defined for the WMQFTE agent. These
     *     values can only be read, they cannot be updated by the
     *     implementation.
     * @return {@code true} if the initialization is successful and {@code
     *     false} if unsuccessful. If {@code false} is returned from an
     *     exit, the exit will not be used.
     */
    boolean initialize(final Map<String, String> agentProperties);

    /**
     * Indicates whether this I/O user exit supports the specified path.
     * <p>
     * This method is used by WMQFTE to determine whether the I/O user exit
     * should be used within a transfer. If no I/O user exit returns true for
     * this method, the default WMQFTE file I/O function will be used.
     *
     * @param path
     *     The path to the required I/O resource.
     * @return {@code true} if the specified path is supported by the I/O exit,
     *     {@code false} otherwise
     */
    boolean isSupported(String path);

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *     The path to the required I/O resource.
     * @return A {@link IOExitPath} instance for the specified path.
     */
}

```

```

* @throws IOException
*         If the path cannot be created for any reason.
*/
IOExitPath newPath(String path) throws IOException;

/**
* Obtains a new {@link IOExitPath} instance for the specified I/O resource
* path and passes record format and length information required by the
* WMQFTE transfer.
* <p>
* Typically this method will be called for the following cases:
* <ul>
* <li>A path where a call to {@link #newPath(String)} has previously
* returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
* re-establishing a new {@link IOExitPath} instance for the path, from an
* internally-serialized state. The passed recordFormat and recordLength
* will be the same as those for the original
* {@link IOExitRecordResourcePath} instance.</li>
* <li>A transfer destination path where the source of the transfer is
* record oriented. The passed recordFormat and recordLength will be the
* same as those for the source.</li>
* </ul>
* The implementation can act on the record format and length information as
* deemed appropriate. For example, for a destination agent if the
* destination does not already exist and the source of the transfer is
* record oriented, the passed recordFormat and recordLength information
* could be used to create an appropriate record-oriented destination path.
* If the destination path already exists, the passed recordFormat and
* recordLength information could be used to perform a compatibility check
* and throw an {@link IOException} if the path is not compatible. A
* compatibility check could ensure that a record oriented path's record
* format is the same as the passed record format or that the record length
* is greater or equal to the passed record length.
* <p>
* This method will be invoked by WMQFTE only if the
* {@link #isSupported(String)} method has been called for the path and
* returned {@code true}.
*
* @param path
*         The path to the required I/O resource.
* @param recordFormat
*         The advised record format.
* @param recordLength
*         The advised record length.
* @return A {@link IOExitPath} instance for the specified path.
* @throws IOException
*         If the path cannot be created for any reason. For example,
*         the passed record format or length is incompatible with the
*         path's actual record format or length.
*/
IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
    throws IOException;

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitChannel.java interface

IOExitChannel.java

```

/*
* Licensed Materials - Property of IBM
*
* "Restricted Materials of IBM"
*
* 5724-H72
*
* © Copyright IBM Corp. 2011, 2024. All Rights Reserved.

```

```

*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

    /**
     * Obtains the data size for the associated {@link IOExitResourcePath} in
     * bytes.
     *
     * @return The data size in bytes.
     * @throws IOException
     *         If a problem occurs while attempting obtain the size.
     */
    long size() throws IOException;

    /**
     * Closes the channel, flushing any buffered write data to the resource and
     * releasing any locks.
     *
     * @throws RecoverableIOException
     *         If a recoverable problem occurs while closing the resource.
     *         This means that WMQFTE can attempt to recover the transfer.
     * @throws IOException
     *         If some other I/O problem occurs. For example, the channel might
     *         already be closed.
     */
    void close() throws RecoverableIOException, IOException;

    /**
     * Reads data from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Data is copied into the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes read.
     *
     * @param buffer
     *         The buffer that the data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *         If a recoverable problem occurs while reading the data. For a
     *         WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *         If some other I/O problem occurs. For a WMQFTE transfer this
     *         means that it will be failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes data to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Data is copied from the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes written.
     *
     * @param buffer
     *         The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     *         If a recoverable problem occurs while writing the data. For a
     *         WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *         If some other I/O problem occurs. For a WMQFTE transfer this
     *         means that it will be failed.
     */
    int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

```

```

/**
 * Forces any updates to this channel's resource to be written to its
 * storage device.
 * <p>
 * This method is required to force changes to both the resource's content
 * and any associated metadata to be written to storage.
 *
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while performing the force.
 *         For a WMQFTE transfer this means that it will attempt to
 *         recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
void force() throws RecoverableIOException, IOException;

/**
 * Attempts to lock the entire resource associated with the channel for
 * shared or exclusive access.
 * <p>
 * The intention is for this method not to block if the lock is currently
 * unavailable.
 *
 * @param shared
 *        {@code true} if a shared lock is required, {@code false} if an
 *        exclusive lock is required.
 * @return A {@link IOExitLock} instance representing the newly acquired
 *         lock or null if the lock cannot be obtained.
 * @throws IOException
 *         If a problem occurs while attempting to acquire the lock.
 */
IOExitLock tryLock(boolean shared) throws IOException;
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitLock.java interface

IOExitLock.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */

```

```

public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *         If the channel associated with the lock is not open or
     *         another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;

    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitPath.java interface

IOExitPath.java

```

/**
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

    /**
     * Obtains the abstract path as a {@link String}.
     */

```

```

*
* @return The abstract path as a {@link String}.
*/
String getPath();

/**
 * Obtains the name portion of this abstract path as a {@link String}.
 * <p>
 * For example, a UNIX-style file system implementation evaluates the
 * path {@code /home/fteuser/file1.txt} as having a name of {@code
 * file1.txt}.
 *
 * @return the name portion of this abstract path as a {@link String}.
 */
String getName();

/**
 * Obtains the parent path for this abstract path as a {@link String}.
 * <p>
 * For example, a UNIX-style file system implementation evaluates the
 * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
 * /home/fteuser}.
 *
 * @return The parent portion of the path as a {@link String}.
 */
String getParent();

/**
 * Obtains the abstract paths that match this abstract path.
 * <p>
 * If this abstract path denotes a directory resource, a list of paths
 * for all resources within the directory are returned.
 * <p>
 * If this abstract path denotes a wildcard, a list of all paths
 * matching the wildcard are returned.
 * <p>
 * Otherwise null is returned, because this abstract path probably denotes a
 * single file resource.
 *
 * @return An array of {@link IOExitResourcePath}s that
 *         match this path, or null if this method is not applicable.
 */
IOExitResourcePath[] listPaths();
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitProperties.java interface

IOExitProperties.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**

```

```

* Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
* aspects of I/O. For example, whether to use intermediate files.
*/
public class IOExitProperties {

    private boolean rereadSourceOnRestart = true;
    private boolean rechecksumSourceOnRestart = true;
    private boolean rechecksumDestinationOnRestart = true;
    private boolean useIntermediateFileAtDestination = true;
    private boolean requiresSingleThreadedChannelIO = false;

    /**
     * Determines whether the I/O exit implementation expects the resource to be
     * re-read from the start if a transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the source
     * resource to be opened at the beginning and re-read from the
     * beginning (the {@link IOExitPath#openForRead(long)} method is
     * always invoked with 0L as an argument). {@code false} if, on
     * restart, the I/O exit expects the source to be opened at the
     * offset that the source agent intends to start reading from (the
     * {@link IOExitPath#openForRead(long)} method can be invoked with a
     * non-zero value as its argument).
     */
    public boolean getRereadSourceOnRestart() {
        return rereadSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation expects
     * the resource to be re-read from the beginning if a transfer is restarted.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rereadSourceOnRestart
     *     {@code true} if, on restart, the I/O exit expects the source
     *     resource to be opened at the beginning and re-read from the
     *     beginning (the {@link IOExitPath#openForRead(long)} method
     *     is always invoked with 0L as an argument). {@code false}
     *     if, on restart, the I/O exit expects the source to be opened
     *     at the offset that the source agent intends to start reading
     *     from (the {@link IOExitPath#openForRead(long)} method can be
     *     invoked with a non-zero value as its argument).
     */
    public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
        this.rereadSourceOnRestart = rereadSourceOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the source
     * resource to be re-checksummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     * {@link #getRereadSourceOnRestart()} method returns {@code true}.
     *
     * @return {@code true} if, on restart, the I/O exit expects the already-
     * transferred portion of the source to be re-checksummed for
     * inconsistencies. Use this option in environments
     * where the source could be changed during a restart. {@code
     * false} if, on restart, the I/O exit does not require the
     * already-transferred portion of the source to be re-checksummed.
     */
    public boolean getRechecksumSourceOnRestart() {
        return rechecksumSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * the source resource to be re-checksummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     * {@link #getRereadSourceOnRestart()} method returns {@code true}.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rechecksumSourceOnRestart
     *     {@code true} if, on restart, the I/O exit expects the already
     *     transferred portion of the source to be re-checksummed
     *     for inconsistencies. Use this option in environments
     *     where the source could be changed during a restart.
     *     {@code false} if, on restart, the I/O exit does not
     *     require the already-transferred portion of the source to be

```

```

*         re-checksummed.
*/
public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
    this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the destination
 * resource to be re-checksummed if the transfer is restarted.
 *
 * @return {@code true} if, on restart, the I/O exit expects the already
 *         transferred portion of the destination to be re-checksummed to
 *         check for inconsistencies. This option should be used in
 *         environments where the destination could have been changed while
 *         a restart is occurring. {@code false} if, on restart, the I/O exit
 *         does not require the already transferred portion of the
 *         destination to be re-checksummed.
 */
public boolean getRechecksumDestinationOnRestart() {
    return rechecksumDestinationOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the destination resource to be re-checksummed if the transfer is
 * restarted.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumDestinationOnRestart
 *        {@code true} if, on restart, the I/O exit expects the already-
 *        transferred portion of the destination to be re-checksummed
 *        for inconsistencies. Use this option in environments
 *        where the destination could have been changed during a
 *        restart. {@code false} if, on restart, the I/O exit does not
 *        require the already-transferred portion of the destination
 *        to be re-checksummed.
 */
public void setRechecksumDestinationOnRestart(
    boolean rechecksumDestinationOnRestart) {
    this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the use of an
 * intermediate file when writing the data at the destination. The
 * intermediate file mechanism is typically used to prevent an incomplete
 * destination resource from being processed.
 *
 * @return {@code true} if data should be written to an intermediate file at
 *         the destination and then renamed (to the requested destination
 *         path name as specified in the transfer request) after the transfer is
 *         complete. {@code false} if data should be written directly to the
 *         requested destination path name without the use of an
 *         intermediate file.
 */
public boolean getUseIntermediateFileAtDestination() {
    return useIntermediateFileAtDestination;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the use of an intermediate file when writing the data at the destination.
 * The intermediate file mechanism is typically used to prevent an
 * incomplete destination resource from being processed.
 *
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param useIntermediateFileAtDestination
 *        {@code true} if data should be written to an intermediate file
 *        at the destination and then renamed (to the requested
 *        destination path name as specified in the transfer request) after
 *        the transfer is complete. {@code false} if data should be written
 *        directly to the requested destination path name without the
 *        use of an intermediate file
 */
public void setUseIntermediateFileAtDestination(
    boolean useIntermediateFileAtDestination) {
    this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
}

```

```

}

/**
 * Determines whether the I/O exit implementation requires
 * {@link IOExitChannel} instances to be accessed by a single thread only.
 *
 * @return {@code true} if {@link IOExitChannel} instances are to be
 *         accessed by a single thread only.
 */
public boolean requiresSingleThreadedChannelIO() {
    return requiresSingleThreadedChannelIO;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * channel operations for a particular instance to be accessed by a
 * single thread only.
 *
 * <p>
 * For certain I/O implementations it is necessary that resource path
 * operations such as open, read, write, and close are invoked only from a
 * single execution {@link Thread}. When set {@code true}, WMQFTE ensures
 * that the following are invoked on a single thread:
 *
 * <ul>
 * <li>{@link IOExitResourcePath#openForRead(long)} method and all methods of
 * the returned {@link IOExitChannel} instance.</li>
 * <li>{@link IOExitResourcePath#openForWrite(boolean)} method and all
 * methods of the returned {@link IOExitChannel} instance.</li>
 * </ul>
 *
 * <p>
 * This has a slight performance impact, hence enable single-threaded channel
 * I/O only when absolutely necessary.
 *
 * <p>
 * The default is {@code false}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param requiresSingleThreadedChannelIO
 *        {@code true} if {@link IOExitChannel} instances are to be
 *        accessed by a single thread only.
 */
public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
    this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
}
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitRecordChannel.java interface

IOExitRecordChannel.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

```

```

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

    /**
     * Reads records from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Record data is copied into the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes read.
     * <p>
     * Only whole records are copied into the buffer.
     * <p>
     * For a fixed-record-format resource, this might be multiple records. The
     * amount of data in the return buffer does not necessarily need to be a
     * multiple of the record length, but the last record is still to be treated
     * as a complete record and padded as required by the caller.
     * <p>
     * For a variable-format resource, this is a single whole record of a size
     * corresponding to the amount of return data or multiple whole records with
     * all except the last being treated as records of maximum size.
     *
     * @param buffer
     *         The buffer that the record data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *         If a recoverable problem occurs while reading the data. For a
     *         WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *         If some other I/O problem occurs, for example, if the passed
     *         buffer is insufficient to contain at least one complete
     *         record). For a WMQFTE transfer this means that it will be
     *         failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes records to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Record data is copied from the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes written.
     * <p>
     * The buffer is expected to contain only whole records.
     * <p>
     * For a fixed-record-format resource, this might be multiple records and if
     * there is insufficient data in the buffer for a complete record, the
     * record is to be padded as required to complete the record.
     * <p>
     * For a variable-record format resource the buffer is normally expected to
     * contain a single record of length corresponding to the amount of data
     * within the buffer. However, if the amount of data within the buffer
     * exceeds the maximum record length, the implementation can either:
     * <ol>
     * <li>throw an {@link IOException} indicating that it cannot handle the
     * situation.</li>
     * <li>Consume a record's worth of data from the buffer, leaving the remaining
     * data within the buffer.</li>
     * <li>Consume all the buffer data and just write what it can to the current
     * record. This effectively truncates the data.</li>
     * <li>Consume all the buffer data and write to multiple records.</li>
     * </ol>
     *
     * @param buffer
     *         The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     */
}

```

```

*           If a recoverable problem occurs while writing the data. For a
*           WMQFTE transfer this means that it will attempt to recover.
* @throws IOException
*           If some other I/O problem occurs. For a WMQFTE transfer this
*           means that it will be failed.
*/
int write(ByteBuffer buffer) throws RecoverableIOException, IOException;
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

 *IOExitRecordResourcePath.java interface*

IOExitRecordResourcePath.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

    /**
     * Record formats for record-oriented resources.
     */
    public enum RecordFormat {
        FIXED, VARIABLE
    }

    /**
     * Obtains the record length for records that are maintained by the resource
     * denoted by this abstract path.
     * <p>
     * For a resource with fixed-length records, the data for each record read
     * and written is assumed to be this length.
     * <p>
     * For a resource with variable-length records, this is the maximum length
     * for a record's data.
     * <p>
     * This method should return a value greater than zero, otherwise it can
     * result in the failure of a WMQFTE transfer that involves this abstract
     * path.
     *
     * @return The record length, in bytes, for records maintained by the
     *         resource.
     */
    int getRecordLength();
}

```

```

/**
 * Obtains record format, as a {@link RecordFormat} instance, for records
 * that are maintained by the resource denoted by this abstract path.
 *
 * @return A {@link RecordFormat} instance for the record format for records
 *         that are maintained by the resource denoted by this abstract
 *         path.
 */
RecordFormat getRecordFormat();

/**
 * Opens a {@link IOExitRecordChannel} instance for reading data from the
 * resource denoted by this abstract path. The current data byte position
 * for the resource is expected to be the passed position value, such that
 * when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
 * data starting from that position is read.
 * <p>
 * Note that the data byte read position will be on a record boundary.
 *
 * @param position
 *         The required data byte read position.
 * @return A new {@link IOExitRecordChannel} instance allowing data to be
 *         read from the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while attempting to open the
 *         resource for reading. This means that WMQFTE can attempt to
 *         recover the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
 */
IOExitRecordChannel openForRead(long position)
    throws RecoverableIOException, IOException;

/**
 * Opens a {@link IOExitRecordChannel} instance for writing data to the
 * resource denoted by this abstract path. Writing of data, using the
 * {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
 * either the beginning of the resource or end of the current data for the
 * resource, depending on the specified append parameter.
 *
 * @param append
 *         When {@code true} indicates that data written to the resource
 *         should be appended to the end of the current data. When
 *         {@code false} indicates that writing of data is to start at
 *         the beginning of the resource; any existing data is lost.
 * @return A new {@link IOExitRecordChannel} instance allowing data to be
 *         written to the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while attempting to open the
 *         resource for writing. This means that WMQFTE can attempt to
 *         recover the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
 */
IOExitRecordChannel openForWrite(boolean append)
    throws RecoverableIOException, IOException;
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitResourcePath.java interface

IOExitResourcePath.java

```

/*
 * Licensed Materials - Property of IBM

```

```

*
* "Restricted Materials of IBM"
*
* 5724-H72
*
* □ Copyright IBM Corp. 2011, 2024. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory\(\)} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile\(\)} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

    /**
     * Creates a new {@link IOExitResourcePath} instance for a child path of the
     * resource denoted by this abstract path.
     * <p>
     * For example, with a UNIX-style path, {@code}
     * IOExitResourcePath\("/home/fteuser/test"\).newPath\("subtest"\) could be
     * equivalent to: {@code IOExitResourcePath\("/home/fteuser/test/subtest"\)}
     *
     * @param child
     *         The child path name.
     * @return A new {@link IOExitResourcePath} instance that represents a child
     *         of this path.
     */
    IOExitResourcePath newPath(final String child);

    /**
     * Creates the directory path for the resource denoted by this abstract
     * path, including any necessary but nonexistent parent directories. If the
     * directory path already exists, this method has no effect.
     * <p>
     * If this operation fails, it might have succeeded in creating some of the
     * necessary parent directories.
     *
     * @throws IOException
     *         If the directory path cannot be fully created, when it does
     *         not already exist.
     */
    void makePath() throws IOException;

    /**
     * Obtains the canonical path of the abstract path as a {@link String}.
     * <p>
     * A canonical path is defined as being absolute and unique. For example,
     * the path can be represented as UNIX-style relative path: {@code}
     * test/file.txt but the absolute and unique canonical path representation
     * is: {@code /home/fteuser/test/file.txt}
     *
     * @return The canonical path as a {@link String}.
     * @throws IOException
     *         If the canonical path cannot be determined for any reason.
     */
    String getCanonicalPath() throws IOException;

    /**
     * Tests if this abstract path is an absolute path.
     * <p>
     * For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
     * path, whereas {@code fteuser/test} is not.
     *
     * @return {@code true} if this abstract path is an absolute path, {@code}
     *         false otherwise.
     */
}

```

```

*/
boolean isAbsolute();

/**
 * Tests if the resource denoted by this abstract path exists.
 *
 * @return {@code true} if the resource denoted by this abstract path
 *         exists, {@code false} otherwise.
 * @throws IOException
 *         If the existence of the resource cannot be determined for any
 *         reason.
 */
boolean exists() throws IOException;

/**
 * Tests whether the calling application can read the resource denoted by
 * this abstract path.
 *
 * @return {@code true} if the resource for this path exists and can be
 *         read, {@code false} otherwise.
 * @throws IOException
 *         If a problem occurs while attempting to determine if the
 *         resource can be read.
 */
boolean canRead() throws IOException;

/**
 * Tests whether the calling application can modify the resource denoted by
 * this abstract path.
 *
 * @return {@code true} if the resource for this path exists and can be
 *         modified, {@code false} otherwise.
 * @throws IOException
 *         If a problem occurs while attempting to determine if the
 *         resource can be modified.
 */
boolean canWrite() throws IOException;

/**
 * Tests whether the specified user is permitted to read the resource
 * denoted by this abstract path.
 *
 * <p>
 * When WMQFTE invokes this method, the user identifier is the MQMD user
 * identifier for the requesting transfer.
 *
 * @param userId
 *         User identifier to test for access.
 * @return {@code true} if the resource for this abstract path exists and is
 *         permitted to be read by the specified user, {@code false}
 *         otherwise.
 * @throws IOException
 *         If a problem occurs while attempting to determine if the user
 *         is permitted to read the resource.
 */
boolean readPermitted(String userId) throws IOException;

/**
 * Tests whether the specified user is permitted to modify the resource
 * denoted by this abstract path.
 *
 * <p>
 * When WMQFTE invokes this method, the user identifier is the MQMD user
 * identifier for the requesting transfer.
 *
 * @param userId
 *         User identifier to test for access.
 * @return {@code true} if the resource for this abstract path exists and is
 *         permitted to be modified by the specified user, {@code false}
 *         otherwise.
 * @throws IOException
 *         If a problem occurs while attempting to determine if the user
 *         is permitted to modify the resource.
 */
boolean writePermitted(String userId) throws IOException;

/**
 * Tests if the resource denoted by this abstract path is a directory-type
 * resource.
 *
 * @return {@code true} if the resource denoted by this abstract path is a
 *         directory type resource, {@code false} otherwise.
 */
boolean isDirectory();

```

```

/**
 * Creates the resource denoted by this abstract path, if it does not
 * already exist.
 *
 * @return {@code true} if the resource does not exist and was successfully
 *         created, {@code false} if the resource already existed.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while attempting to create
 *         the resource. This means that WMQFTE can attempt to recover
 *         the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
 */
boolean createNewPath() throws RecoverableIOException, IOException;

/**
 * Tests if the resource denoted by this abstract path is a file-type
 * resource.
 *
 * @return {@code true} if the resource denoted by this abstract path is a
 *         file type resource, {@code false} otherwise.
 */
boolean isFile();

/**
 * Obtains the last modified time for the resource denoted by this abstract
 * path.
 * <p>
 * This time is measured in milliseconds since the epoch (00:00:00 GMT,
 * January 1, 1970).
 *
 * @return The last modified time for the resource denoted by this abstract
 *         path, or a value of 0L if the resource does not exist or a
 *         problem occurs.
 */
long lastModified();

/**
 * Deletes the resource denoted by this abstract path.
 * <p>
 * If the resource is a directory, it must be empty for the delete to work.
 *
 * @throws IOException
 *         If the delete of the resource fails for any reason.
 */
void delete() throws IOException;

/**
 * Renames the resource denoted by this abstract path to the specified
 * destination abstract path.
 * <p>
 * The rename should still be successful if the resource for the specified
 * destination abstract path already exists and it is possible to replace
 * it.
 *
 * @param destination
 *         The new abstract path for the resource denoted by this
 *         abstract path.
 * @throws IOException
 *         If the rename of the resource fails for any reason.
 */
void renameTo(IOExceptionResourcePath destination) throws IOException;

/**
 * Creates a new path to use for writing to a temporary resource that did
 * not previously exist.
 * <p>
 * The implementation can choose the abstract path name for the temporary
 * resource. However, for clarity and problem diagnosis, the abstract path
 * name for the temporary resource should be based on this abstract path
 * name with the specified suffix appended and additional characters to make
 * the path unique (for example, sequence numbers), as required.
 * <p>
 * When WMQFTE transfers data to a destination it normally attempts to first
 * write to a temporary resource then on transfer completion renames the
 * temporary resource to the required destination. This method is called by
 * WMQFTE to create a new temporary resource path. The returned path should
 * be new and the resource should not previously exist.
 *
 * @param suffix
 *         Recommended suffix to use for the generated temporary path.

```

```

*
* @return A new {@link IOExitResourcePath} instance for the temporary
*         resource path, that did not previously exist.
* @throws RecoverableIOException
*         If a recoverable problem occurs whilst attempting to create
*         the temporary resource. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitResourcePath createTempPath(String suffix)
    throws RecoverableIOException, IOException;

/**
 * Opens a {@link IOExitChannel} instance for reading data from the resource
 * denoted by this abstract path. The current data byte position for the
 * resource is expected to be the passed position value, such that when
 * {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
 * from that position is read.
 *
 * @param position
 *         The required data byte read position.
 * @return A new {@link IOExitChannel} instance allowing data to be read
 *         from the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while attempting to open the
 *         resource for reading. This means that WMQFTE can attempt to
 *         recover the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
*/
IOExitChannel openForRead(long position) throws RecoverableIOException,
    IOException;

/**
 * Opens a {@link IOExitChannel} instance for writing data to the resource
 * denoted by this abstract path. Writing of data, using the
 * {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
 * the beginning of the resource or end of the current data for the
 * resource, depending on the specified append parameter.
 *
 * @param append
 *         When {@code true} indicates that data written to the resource
 *         should be appended to the end of the current data. When
 *         {@code false} indicates that writing of data is to start at
 *         the beginning of the resource; any existing data is lost.
 * @return A new {@link IOExitChannel} instance allowing data to be written
 *         to the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs whilst attempting to open the
 *         resource for writing. This means that WMQFTE can attempt to
 *         recover the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
*/
IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
    IOException;

/**
 * Tests if the resource denoted by this abstract path is in use by another
 * application. Typically, this is because another application has a lock on
 * the resource either for shared or exclusive access.
 *
 * @return {@code true} if resource denoted by this abstract path is in use
 *         by another application, {@code false} otherwise.
*/
boolean inUse();

/**
 * Obtains a {@link IOExitProperties} instance for properties associated
 * with the resource denoted by this abstract path.
 *
 * <p>
 * WMQFTE will read these properties to govern how a transfer behaves when
 * interacting with the resource.
 *
 * @return A {@link IOExitProperties} instance for properties associated
 *         with the resource denoted by this abstract path.
*/
IOExitProperties getProperties();
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

IOExitWildcardPath.java interface

IOExitWildcardPath.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {
```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“Using MFT transfer I/O user exits” on page 249](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

MonitorExit.java interface

MonitorExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2009, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
```

```

* user exit routine processing. This interface defines a method that will be
* invoked immediately prior to starting a task as the result of a monitor trigger
*/
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constant
     *      defined in EnvironmentMetaDataConstants class can
     *      be used to access the data held by this map.
     *
     * @param monitorMetaData
     *      meta data to associate with the monitor. The meta data passed
     *      to this method can be altered, and the changes will be
     *      reflected in subsequent exit routine invocations. This map
     *      also contains keys with IBM reserved names. These entries are
     *      defined in the MonitorMetaDataConstants class and
     *      have special semantics. The the values of the IBM reserved names
     *      cannot be modified by the exit
     *
     * @param taskDetails
     *      An XML String representing the task to be executed as a result of
     *      the monitor triggering. This XML string may be modified by the
     *      exit
     *
     * @return
     *      a monitor exit result object which is used to determine if the
     *      task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                               Map<String, String> monitorMetaData,
                               Reference<String> taskDetails);
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related tasks

[“Monitoring MFT resources” on page 162](#)

You can monitor Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the **Monitors** view in the Managed File Transfer plug-in for IBM MQ Explorer.

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“SourceTransferEndExit.java interface” on page 883](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

ProtocolBridgeCredentialExit.java interface

ProtocolBridgeCredentialExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.

```

```

*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user ID of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *       The values of properties defined for the protocol bridge.
     *       These values can only be read, they cannot be updated by
     *       the implementation.
     *
     * @return true if the initialization is successful and false if unsuccessful
     *         If false is returned from an exit the protocol bridge agent will not
     *         start
     */
    public boolean initialize(final Map<String> bridgeProperties);

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer message to the
     * credentials to be used to access the protocol server
     *
     * @param mqUserId The MQ user ID from which to map to the credentials to be used
     *                 access the protocol server
     * @return A credential exit result object that contains the result of the map and
     *         the credentials to use to access the protocol server
     */
    public CredentialExitResult mapMQUserId(final String mqUserId);

    /**
     * Invoked once when a protocol bridge agent is shutdown. It is intended to release
     * any resources that were allocated by the exit
     *
     * @param bridgeProperties
     *       The values of properties defined for the protocol bridge.
     *       These values can only be read, they cannot be updated by
     *       the implementation.
     *
     * @return
     */
    public void shutdown(final Map<String> bridgeProperties);
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related tasks

[“Mapping credentials for a file server by using exit classes” on page 214](#)

If you do not want to use the default credential mapping function of the protocol bridge agent, you can map user credentials in Managed File Transfer to user credentials on the file server by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping. If you configure credential mapping user exits, they take the place of the default credential mapping function.

ProtocolBridgeCredentialExit2.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit2 extends
    ProtocolBridgeCredentialExit {

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer
     * message to the credentials used to access a specified protocol server.
     *
     * @param endPoint
     *     Information that describes the protocol server to be accessed.
     * @param mqUserId
     *     The MQ user ID from which to map the credentials used to
     *     access the protocol server.
     * @return A {@link CredentialExitResult} instance that contains the result
     *     of the map and the credentials to use to access the protocol
     *     server.
     */
    public CredentialExitResult mapMQUserId(
        final ProtocolServerEndPoint endPoint, final String mqUserId);
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related tasks

[“Mapping credentials for a file server by using exit classes” on page 214](#)

If you do not want to use the default credential mapping function of the protocol bridge agent, you can map user credentials in Managed File Transfer to user credentials on the file server by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping. If you configure credential mapping user exits, they take the place of the default credential mapping function.

ProtocolBridgePropertiesExit2.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2024. All Rights Reserved.

```

```

*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *     The values of properties defined for the protocol bridge.
     *     These values can only be read, they cannot be updated by the
     *     implementation.
     * @return {@code true} if the initialization is successful and {@code
     *     false} if unsuccessful. If {@code false} is returned from an exit
     *     the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
     *
     * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
     */
    public String getCredentialLocation ();

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *     The name of the protocol server whose properties are to be
     *     returned. If a null or a blank value is specified, properties
     *     for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *     if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
        final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *     The values of properties defined for the protocol bridge.
     *     These values can only be read, they cannot be updated by the
     *     implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);
}

```

Related concepts

[“ProtocolBridgePropertiesExit: Looking up protocol file server properties” on page 211](#)

If you have a large number of protocol file servers, you can implement the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit` interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in

preference to maintaining a `ProtocolBridgeProperties.xml` file. You are recommended to use the `ProtocolBridgePropertiesExit2.java` interface but the `ProtocolBridgePropertiesExit.java` interface is also supported. If you have an existing implementation of the `ProtocolBridgePropertiesExit.java` interface from IBM WebSphere MQ File Transfer Edition, you can use it in IBM WebSphere MQ 7.5 or later. The new `getCredentialLocation` method in `ProtocolBridgePropertiesExit2.java` uses the default location of the `ProtocolBridgeCredentials.xml` file, which is your home directory.

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related tasks

[“Mapping credentials for a file server by using exit classes” on page 214](#)

If you do not want to use the default credential mapping function of the protocol bridge agent, you can map user credentials in Managed File Transfer to user credentials on the file server by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping. If you configure credential mapping user exits, they take the place of the default credential mapping function.

SourceFileExitFileSpecification.java class

SourceFileExitFileSpecification.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2012, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor. Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *        the source file specification to associate with the source file
     *        exit file specification.
     *
     * @param destinationFileSpecification
     *        the destination file specification to associate with the
     *        source file exit file specification.
     *
     * @param sourceFileMetaData
     *        the source file meta data.
     *
     * @param destinationFileMetaData
     *        the destination file meta data
     */
    public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                           final String destinationFileSpecification,
                                           final Map<String, String> sourceFileMetaData,
                                           final Map<String, String> destinationFileMetaData) {
        this.sourceFileSpecification = sourceFileSpecification;
        this.destinationFileSpecification = destinationFileSpecification;
    }
}
```

```

        this.sourceFileMetaData = sourceFileMetaData;
        this.destinationFileMetaData = destinationFileMetaData;
    }

    /**
     * Returns the destination file specification.
     *
     * @return the destination file specification. This represents the location,
     *         on the agent acting as the destination for the transfer, where the
     *         file should be written. Exit routines installed into the agent
     *         acting as the destination for the transfer may override this value.
     */
    public String getDestination() {
        return destinationFileSpecification;
    }

    /**
     * Returns the source file specification.
     *
     * @return the source file specification. This represents the location where
     *         the file data will be read from.
     */
    public String getSource() {
        return sourceFileSpecification;
    }

    /**
     * Returns the file meta data that relates to the source file specification.
     *
     * @return the file meta data that relates to the source file specification.
     */
    public Map<String, String> getSourceFileMetaData() {
        return sourceFileMetaData;
    }

    /**
     * Returns the file meta data that relates to the destination file specification.
     *
     * @return the file meta data that relates to the destination file specification.
     */
    public Map<String, String> getDestinationFileMetaData() {
        return destinationFileMetaData;
    }
}

```

Related concepts

“Metadata for MFT user exits” on page 845

There are three different types of metadata that can be supplied to user exit routines for Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

SourceTransferEndExit.java interface

SourceTransferEndExit.java

```

/**
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
 */

```

```

public interface SourceTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param transferExitResult
     *     a result object reflecting whether or not the transfer completed
     *     successfully.
     *
     * @param sourceAgentName
     *     the name of the agent acting as the source of the transfer.
     *     This is the name of the agent that the implementation of this
     *     method will be invoked from.
     *
     * @param destinationAgentName
     *     the name of the agent acting as the destination of the
     *     transfer.
     *
     * @param environmentMetaData
     *     meta data about the environment in which the implementation
     *     of this method is running. This information can only be read,
     *     it cannot be updated by the implementation. The constants
     *     defined in EnvironmentMetaDataConstants class can
     *     be used to access the data held by this map.
     *
     * @param transferMetaData
     *     meta data to associate with the transfer. The information can
     *     only be read, it cannot be updated by the implementation. This
     *     map may also contain keys with IBM reserved names. These
     *     entries are defined in the TransferMetaDataConstants
     *     class and have special semantics.
     *
     * @param fileResults
     *     a list of file transfer result objects that describe the source
     *     file name, destination file name and result of each file transfer
     *     operation attempted.
     *
     * @return
     *     an optional description to enter into the log message describing
     *     transfer completion. A value of null can be used
     *     when no description is required.
     */
    String onSourceTransferEnd(TransferExitResult transferExitResult,
                               String sourceAgentName,
                               String destinationAgentName,
                               Map<String, String>environmentMetaData,
                               Map<String, String>transferMetaData,
                               List<FileTransferResult>fileResults);

}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

SourceTransferStartExit.java interface

SourceTransferStartExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"

```

```

*
* 5724-H72
*
*   Copyright IBM Corp. 2008, 2024. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param sourceAgentName
     *     the name of the agent acting as the source of the transfer.
     *     This is the name of the agent that the implementation of this
     *     method will be invoked from.
     *
     * @param destinationAgentName
     *     the name of the agent acting as the destination of the
     *     transfer.
     *
     * @param environmentMetaData
     *     meta data about the environment in which the implementation
     *     of this method is running. This information can only be read,
     *     it cannot be updated by the implementation. The constants
     *     defined in EnvironmentMetaDataConstants class can
     *     be used to access the data held by this map.
     *
     * @param transferMetaData
     *     meta data to associate with the transfer. The meta data passed
     *     to this method can be altered, and the changes to will be
     *     reflected in subsequent exit routine invocations. This map may
     *     also contain keys with IBM reserved names. These entries are
     *     defined in the TransferMetaDataConstants class and
     *     have special semantics.
     *
     * @param fileSpecs
     *     a list of file specifications that govern the file data to
     *     transfer. The implementation of this method can add entries,
     *     remove entries, or modify entries in this list and the changes
     *     will be reflected in the files transferred.
     *
     * @return
     *     a transfer exit result object which is used to determine if the
     *     transfer should proceed, or be cancelled.
     */
    TransferExitResult onSourceTransferStart(String sourceAgentName,
        String destinationAgentName,
        Map<String, String> environmentMetaData,
        Map<String, String> transferMetaData,
        List<SourceFileExitFileSpecification> fileSpecs);
}

```

Related concepts

[“Customizing MFT with user exits” on page 245](#)

You can customize the features of Managed File Transfer by using your own programs known as user exit routines.

Related reference

[“SourceFileExitFileSpecification.java class” on page 882](#)

[“SourceTransferEndExit.java interface” on page 883](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

TransferExitResult.java interface

TransferExitResult.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a transfer exit routine. It is composed of a result
 * code, which determines if the transfer should proceed, and an optional explanatory
 * message. The explanation, if present, is entered into the log message.
 */
public class TransferExitResult {

    private final TransferExitResultCode resultCode;
    private final String explanation;

    /**
     * For convenience, a static "proceed" result with no associated explanation
     * message.
     */
    public static final TransferExitResult PROCEED_RESULT =
        new TransferExitResult(TransferExitResultCode.PROCEED, null);

    /**
     * Constructor. Creates a transfer exit result object with a specified result
     * code and explanation.
     *
     * @param resultCode
     *         The result code to associate with the exit result being created.
     *
     * @param explanation
     *         The explanation to associate with the exit result being created.
     *         A value of <code>null</code> can be specified to indicate no
     *         explanation.
     */
    public TransferExitResult(TransferExitResultCode resultCode, String explanation) {
        this.resultCode = resultCode;
        this.explanation = explanation;
    }

    /**
     * Returns the explanation associated with this transfer exit result.
     *
     * @return
     *         the explanation associated with this exit result.
     */
    public String getExplanation() {
        return explanation;
    }

    /**
     * Returns the result code associated with this transfer exit result.
     *
     * @return
     *         the result code associated with this exit result.
     */
    public TransferExitResultCode getResultCode() {
        return resultCode;
    }
}
```

Related tasks

[Customizing MFT with user exits](#)

Related reference

[“SourceTransferStartExit.java interface” on page 884](#)

[“DestinationTransferStartExit.java interface” on page 858](#)

[“DestinationTransferEndExit.java interface” on page 857](#)

[“MonitorExit.java interface” on page 877](#)

[“ProtocolBridgeCredentialExit.java interface” on page 878](#)

Message formats for messages you can put on the MFT Agent command queue

The following XML schemas define the formats for messages that can be put on the agent command queue to request that the agent perform an action. The XML message can be placed on the agent command queue by using the command-line commands or by an application.

File transfer request message format

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

File transfer messages can have one of following three root elements:

- `<request>` - for new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
- `<cancel>` - for canceling file transfers in progress
- `<transferSpecifications>` - for specifying multiple transfer file groups, used by the **fteCreateTransfer** command

For information about specifying multiple transfer groups by using the `<transferSpecifications>` element, see [Using transfer definition files](#).

Schema

The following schema describes which elements are valid in a transfer request XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
<xsd:include schemaLocation="fteutils.xsd"/>
  <!--
    Defines the request of a managed transfer and version number
    <request version="1.00" ...
      <managedTransfer>
        ...
      </managedTransfer>
    </request>
  -->
<xsd:element name="request">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="managedTransfer" type="managedTransferType"/>
      <xsd:element name="deleteScheduledTransfer" type="deleteScheduledTransferType"/>
      <xsd:element name="managedCall" type="managedCallType"/>
    </xsd:choice>
    <xsd:attribute name="version" type="versionType" use="required"/>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

```

</xsd:element>
<!--
  Defines the cancel request of a managed transfer and version number
  <cancel version="1.00"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <originator>
      <hostName>myMachine</hostName>
      <userID>myUserId</userID>
    </originator>      - Delete a scheduled transfer.

    <transfer>
      Transfer ID to Cancel
    </transfer>
  </cancel>
-->
<xsd:element name="cancel">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="originator" type="hostUserIDType" maxOccurs="1" minOccurs="1"/>
      <xsd:choice>
        <xsd:element name="transfer" type="IDType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="call" type="IDType" maxOccurs="1" minOccurs="1"/>
      </xsd:choice>
      <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required"/>
  </xsd:complexType>
</xsd:element>

<!--
  Defines the transfer definition element structure.
  <transferSpecifications>
    <item ...
    <item ...
  </transferSpecifications>
-->
<xsd:element name="transferSpecifications">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="item" type="itemType" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!--
  Define a managed transfer of an instigator and request
  <managedTransfer>

    <originator>
      ...
    </originator>

    <schedule>
      <submit timebase="source"|"UTC">2008-12-07T16:07</submit>
      <repeat>
        <frequency interval="hours">2</frequency>
        <expireTime>2008-12-0816:07</expireTime>
      </repeat>
    </schedule>

    <sourceAgent agent="here" QMgr="near"/>
    <destinationAgent agent="there" QMgr="far"/>

    <trigger>
      ...
    </trigger>

    <transferSet>
      ...
    </transferSet>
  </managedTransfer>
-->

<xsd:complexType name="managedTransferType">
  <xsd:sequence>
    <xsd:element name="originator" type="origTransferRequestType" maxOccurs="1"
minOccurs="1"/>
    <xsd:element name="schedule" type="scheduleType" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="destinationAgent" type="agentClientType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>

```

```

        <xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    This is a modified form of origRequestType which is used on a managed transfer request.
    The hostName and userID are mandatory attributes in this case.
-->
<xsd:complexType name="origTransferRequestType">
    <xsd:sequence>
        <xsd:element name="hostName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="userID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the transferset of source and destination agent and one or more files
    <transferset priority="1">
        <metaDataSet>
            <metaData key="keyname">keyvalue</metaData>
            <metaData key="keyname">keyvalue</metaData>
        </metaDataSet>

        <item>
            ...
        </item>
    </transferset>
-->
<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="preSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="postSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="preDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="postDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="item" type="itemType" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional"/>
</xsd:complexType>

<!--
    Define a file pair with source and destination
    <item mode=[binary|text]>
        <source recursive="false" disposition="leave">
            <file>filename</file>
        </source>

        <destination type="file" exist="error">
            <file>filename</file>
        </destination>

    </item>
-->
<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source" type="fileSourceType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destination" type="fileDestinationType" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required"/>
    <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required"/>
</xsd:complexType>

<!--
    Defines the request to delete scheduled file transfer.
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>
            </delete>
        </originator>

```

```

        <ID>56</ID>
    </deleteScheduledTransfer>
-->
<xsd:complexType name="deleteScheduledTransferType">
    <xsd:sequence>
        <xsd:element name="originator" type="origDeleteType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="ID" type="idType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="managedCallType">
    <xsd:sequence>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="callTransferSetType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="callTransferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="call" type="commandActionType" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional"/>
</xsd:complexType>
</xsd:schema>

```

Understanding the transfer request message

The elements and attributes used in transfer request messages are described in the following list:

Element descriptions

<request>

Group element containing all the elements required to specify a file transfer request.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<managedTransfer>

Group element that contains all the elements required for a single file transfer or single group of file transfers.

<deleteScheduledTransfer>

Group element that contains originator and ID information to cancel a schedule transfer.

<managedCall>

Group element that contains all the elements required for a single managed call of a program or executable.

<ID>

Unique identifier that specifies the transfer request to delete from the list of pending scheduled transfers.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<schedule>

Group element describing the scheduled time for the file transfer, the repeat behavior, and when the next occurrence is due.

<submit>

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. This attribute can have one of the following values: <ul style="list-style-type: none"> • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time
timezone	The time zone description according to the timebase value

<repeat>

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

<frequency>

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> • minutes • hours • days • weeks • months • years

<expireTime>

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

<expireCount>

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.
hostName	The host name or IP address of the agent queue manager.
portNumber	The port number used for client connections to the destination agent queue manager.
channel	The channel name used to connect to the destination agent queue manager.

<trigger>

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The valid values are as follows: <ul style="list-style-type: none"> • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

<fileExist>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileSize> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid values are as follows: <ul style="list-style-type: none"> • = at least one file name in the name list must match • != a minimum of one of the files in the name list does not exist
value	Indicates the comparison type: <ul style="list-style-type: none"> • exist: file must exist

<fileSize>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileExist> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid value is as follows: <ul style="list-style-type: none"> • >= one of the file names in the name list exists and has a minimum size as specified in the value attribute
value	File size specified as an integer value with units specified as one of the following: <ul style="list-style-type: none"> • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes (the units value is not case-sensitive)

<reply>

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file or the default of `WMQFTE.*` if not specified.

Attribute	Description
detailed	Whether detailed transfer result information is required in the reply message. Multiple reply messages for each transfer can be generated. The valid values are as follows: <ul style="list-style-type: none">• true - detailed reply information is required. The format of the information is the same as that published to the transfer log in the progress messages, that is, the <code><transferSet></code> element. For more information, see “File transfer log message formats” on page 723. Detailed reply information is present only when the transfer source agent has the <code>enableDetailedReplyMessages</code> property set to true.• false - detailed reply information is not required. The default value is false.
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.
persistent	Whether the message written to the reply queue is persistent. The valid values are as follows: <ul style="list-style-type: none">• true - the message is persistent• false - the message is not persistent• qdef - the persistence of the message is defined by the properties of the reply queue The default value is false.

<transferSet>

Specifies a group of file transfers you want to perform together or a group of managed calls that you want to perform together. During transmission `<transferSet>` is a group element containing `<item>` elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

<metaDataSet>

Optional group element containing one or more metadata items.

<metaData>

Specifies the user-defined metadata that is passed to the exit points called by the agent. The element contains the metadata value as a string.

Attribute	Description
key	Metadata name as a string

<call>

Group element that contains `<command>` elements specifying the program or executable to call.

<command>

Specifies the program or executable to call. The command must be located on the agent command path. For more information, see [Table 45 on page 528](#). This element can contain optional <argument> elements.

Attribute	Description
name	The name of the command.
successRC	The successful return code that this command returns. Default is 0.
retryCount	The number of times that the command is to be retried if it fails.
retryWait	The time, in seconds, to wait between retries of the command.
type	The type of program to be called. The valid values are antscript, jcl, or executable.

<argument>

Specifies an argument to pass to the command.

<item>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. The valid values are MD5 or none.

<source>

Group element that specifies files on the source system and whether they are removed after the transfer completes

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid values are as follows: <ul style="list-style-type: none"> • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

<file>

Specifies the transfer source. **Multi** For Multiplatforms, the transfer source can be a file or a directory name. For the z/OS platform, the transfer source can be a file, directory, data set, or PDS name. Use the fully qualified path in the format consistent with your operating system, for example C:/from/here.txt. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the source file. This alias is the name of the source file, excluding any directory path specified for the transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: <ul style="list-style-type: none"> • LF - line feed character only • CRLF - carriage return and line feed character sequence

Attribute	Description
encoding	The encoding of the source file for a text file transfer.
 delimiter	Specifies the delimiter that is included between records in record-oriented source files, for example, z/OS data sets. Specify the delimiter value as two hexadecimal digits in the range 00-FF, prefixed by x. For example, x12 or x03,x7F.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values is as follows: <ul style="list-style-type: none"> • binary - a hexadecimal delimiter This attribute is available only if you have enabled the 7.0.4.1 function.
delimiterPosition	Specifies the position to insert delimiters when writing record-oriented source file records to a normal file. The valid values are as follows: <ul style="list-style-type: none"> • prefix - the delimiter is inserted into the destination file before the data from each source record-oriented file record. • postfix - the delimiter is inserted into the destination file after the data from each source record-oriented file record.
includeDelimiterInFile	Specifies whether to include a delimiter between records in record-oriented source files.
 keepTrailingSpaces	Specifies whether trailing spaces are to be kept on source records read from a fixed-length-format data set as part of a text mode transfer. The default is that trailing spaces are stripped. The valid values are as follows: <ul style="list-style-type: none"> • true - trailing spaces are kept on source records read from a fixed-length-format data set • false - trailing spaces are stripped from source records read from a fixed-length-format data set

<queue>

When used with the <source> element, specifies the name of the queue to transfer from, which must be located on the source agent queue manager. Use the format *QUEUE*. Do not include the queue manager name, the queue must be present on the source agent queue manager. You cannot use the <queue> element inside the <source> element, if you have used it inside of the <destination> element.

Attribute	Description
useGroups	Specifies whether to transfer only the first complete group of messages from the source queue. The valid values are as follows: <ul style="list-style-type: none"> • true - transfer only the first complete group of messages • false - transfer all messages on the source queue
groupId	Specifies the group of messages to read from the source queue. This attribute is valid only when the value of the useGroups attribute is true.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values are as follows: <ul style="list-style-type: none"> • text - a text or Java literal delimiter • binary - a hexadecimal delimiter

Attribute	Description
delimiter	Specifies the delimiter that is included in the destination file between individual message data.
delimiterPosition	Specifies whether the delimiter is included in the destination file before or after individual message data. The valid values are as follows: <ul style="list-style-type: none"> • prefix - the delimiter is included before the data • postfix - the delimiter is included after the data
encoding	Specifies the source queue encoding.
waitTime	Specifies the time, in seconds, for the source agent to wait for either: <ul style="list-style-type: none"> • a message to appear on the source queue, if the queue is empty or has become empty • a complete group to appear on the source queue, if the useGroups attribute has been set to true <p>For information about setting the waitTime value, see “Guidance for specifying a wait time on a message-to-file transfer” on page 657.</p>

<destination>

Group element that specifies the destination and the behavior if files exist at the destination agent.

You can specify only one of <file> and <queue> as a child element of destination.

Attribute	Description
type	<p>The type of destination. The valid values are as follows:</p> <ul style="list-style-type: none"> • file - specifies a file as the destination • directory - specifies a directory as the destination •  dataset - specifies a z/OS data set as the destination •  pds - specifies a z/OS partitioned data set as the destination • queue - specifies an IBM MQ queue as the destination • filespace - specifies a file space as the destination <p>The value queue is valid only when the <destination> element has a child element of <queue>.</p> <p>The value filespace is valid only when the <destination> element has a child element of <filespace>.</p> <p>The other values are valid only when the <destination> element has a child element of <file>.</p>
exist	<p>Specifies the action that is taken if a destination file exists on the destination system. The valid values are as follows:</p> <ul style="list-style-type: none"> • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. <p>This attribute is not valid if the <destination> element has a child element of <queue> or <filespace>.</p>

<file>

Specifies additional settings for the previously-described **<destination>** element. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the <destination> file. This alias is the name of the source file, excluding any directory path specified for the transfer.
encoding	The encoding of the <destination> file for a text file transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: <ul style="list-style-type: none"> • LF - line feed character only • CRLF - carriage return and line feed character sequence
truncateRecords	Optional. Specifies that <destination> records longer than the LRECL data set attribute are truncated. <ul style="list-style-type: none"> • True - the records are truncated • False - the records are wrapped The default setting is false.

<queue>

When used with the **<destination>** element, specifies the name of the queue to transfer to, which can be located on any queue manager that is connected to the destination agent queue manager. Use the format `QUEUE@QM` where `QUEUE` is the name of the queue to put the messages on and `QM` is the queue manager where the queue is located. You cannot use the **<queue>** element inside the **<destination>** element, if you have used it inside of the **<source>** element.

Attribute	Description
delimiter	The delimiter to split the file into multiple messages.
delimiterType	Specifies the type of delimiter. The valid values are as follows: <ul style="list-style-type: none"> • text - a Java regular expression • binary - a sequence of hexadecimal bytes • size - a number of bytes, kibibytes, or mebibytes. For example, 1 B, 1 K, or 1 M.
delimiterPosition	Specifies whether the delimiter is expected before or after the data to include in individual messages. The valid options are as follows: <ul style="list-style-type: none"> • prefix - the delimiter is expected before the data • postfix - the delimiter is expected after the data
includeDelimiterInMessage	A boolean specifying whether to include the delimiters that were used to split the file into multiple messages at the end of the messages.
encoding	Specifies the destination queue encoding.
persistent	Specifies whether the messages are persistent. The valid values are as follows: <ul style="list-style-type: none"> • true - the messages are persistent • false - the messages are not persistent

Attribute	Description
	<ul style="list-style-type: none"> qdef - the persistence value of the messages is defined by the settings on the destination queue
setMqProps	A boolean specifying whether IBM MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
unrecognisedCodePage	<p>Specifies whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager. The valid values are as follows:</p> <ul style="list-style-type: none"> fail - the transfer reports a failure binary - the data is converted to the destination code page and the IBM MQ message header describing the format of the data is set to MQFMT_NONE. <p>The default behavior is fail.</p>

<filespace>

Group element specifying the name of the file space to transfer to.

<name>

When used with the <filespace> element, the value of this element specifies the name of the file space.

<preSourceCall>

Group element specifying a command to call at the source of the transfer, before the transfer starts.

<postSourceCall>

Group element specifying a command to call at the source of the transfer, after the transfer completes.

<preDestinationCall>

Group element specifying a command to call at the destination of the transfer, before the transfer starts.

<postDestinationCall>

Group element specifying a command to call at the destination of the transfer, after the transfer completes.

<command>

When used with the <preSourceCall>, <postSourceCall>, <preDestinationCall>, or <postDestinationCall> element, this element specifies the command to be called. The command must be located on the agent command path. For more information, see [Table 45 on page 528](#).

Attribute	Description
name	The name of the command to run.
successRC	The return code that is expected if the command runs successfully.

<argument>

When used with the <command> element, this element specifies an argument to be passed in to the command. You can have any number of <argument> elements inside a <command> element.

<job>

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: [“File transfer log message formats” on page 723](#).

<name>

When used with the <job> element, the value of this element specifies the name of the job.

<transferSpecifications>

Group element that contains <item> elements for multiple transfer groups. See [Using transfer definition files](#) for further details about how to use this element.

<cancel>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<transfer>

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>

Group element containing job information.

<jobName>

Specifies logical job identifier.

File transfer cancel message format

A file transfer request returns a 48-character ID that identifies the transfer for a specific agent. This ID is used to cancel transfers.

Understanding the transfer cancel message

The elements and attributes used in transfer cancel messages are described:

<cancel>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<transfer>

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>

Optional. Group element containing job information.

<jobName>

Specifies logical job identifier.

Examples

Examples of XML messages that conform to this schema are provided for each of the following requests:

- [Create a file transfer](#)
- [Create an asynchronous file transfer request](#)
- [Cancel a file transfer](#)
- [Create a scheduled transfer](#)
- [Delete a scheduled transfer](#)
- [Create a managed call](#)
- [Create a file transfer that includes managed calls](#)

File transfer request message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

Create transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
version="4.00"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <destinationAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <transferSet>
      <item checksumMethod="MD5" mode="binary">
        <source disposition="leave" recursive="false">
          <file>/etc/passwd</file>
        </source>
        <destination exist="overwrite" type="directory">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Create transfer request - synchronous

When a user requests a blocking synchronous request, that is, they wait for the transfer to complete and receive status messages, the message placed on the command queue contains a reply element that specifies the queue that a reply message is sent to. The following example shows the message placed on the command queue used by FTEAGENT:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="FTEAGENT"
      QMgr="QM1"/>
    <destinationAgent agent="AGENT2"
      QMgr="QM2"/>
    <reply QMGR="QM1">WMQFTE.492D0D5502770020</reply>
```

```

<transferSet>
  <item mode="binary" checksumMethod="MD5">
    <source recursive="false" disposition="leave">
      <file>c:\sourcefiles\source1.doc</file>
    </source>
    <destination type="file" exist="overwrite">
      <file>c:\destinationfiles\dest1.doc</file>
    </destination>
  </item>
</transferSet>
</managedTransfer>
</request>

```

The <reply> element is populated with the name of the command queue manager where a temporary dynamic queue has been created to receive reply about the successful (or otherwise) completion of the transfer. The name of the temporary dynamic queue is composed of two parts:

- The prefix as defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file (it is WMQFTE. by default)
- The ID of the queue as generated by IBM MQ

Cancel transfer request

```

<?xml version="1.0" encoding="UTF-8"?>
<cancel xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <transfer>414D51205553322E42494E444494E47538B0F404D032C0020</transfer>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20002007</reply>
</cancel>

```

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

Scheduled file transfer message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

Create scheduled transfer

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00</submit>
    </schedule>
    <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
    <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
    <transferSet>
      <item checksumMethod="MD5" mode="binary">
        <source disposition="leave" recursive="false">
          <file>/etc/passwd</file>
        </source>

```

```

        <destination exist="overwrite" type="directory">
            <file>/tmp</file>
        </destination>
    </item>
</transferSet>
</managedTransfer>
</request>

```

Delete scheduled transfer

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    version="4.00"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>example.com.</hostName>
                <userID>mqm</userID>
            </delete>
        </originator>
        <ID>1</ID>
        <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003902</reply>
    </deleteScheduledTransfer>
</request>

```

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

MFT agent call request message examples

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

Managed call request example

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    version="1.00"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedCall>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <agent agent="DNWE" QMgr="QM1"/>
        <transferSet>
            <call>
                <command name="echo" successRC="0">
                    <argument>call</argument>
                    <argument>test</argument>
                </command>
            </call>
        </transferSet>
    </job>
    <name>managedCallCalls.xml</name>
</job>
</managedCall>
</request>

```

Managed transfer request example with calls

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent agent="DNWE" QMgr="QM1" />
    <destinationAgent agent="DNWE" QMgr="QM1" />
    <transferSet>
      <preSourceCall>
        <command name="echo" successRC="0">
          <argument>preSourceCall</argument>
          <argument>test</argument>
        </command>
      </preSourceCall>
      <postSourceCall>
        <command name="echo" successRC="0">
          <argument>postSourceCall</argument>
          <argument>test</argument>
        </command>
      </postSourceCall>
      <preDestinationCall>
        <command name="echo" successRC="0">
          <argument>preDestinationCall</argument>
          <argument>test</argument>
        </command>
      </preDestinationCall>
      <postDestinationCall>
        <command name="echo" successRC="0">
          <argument>postDestinationCall</argument>
          <argument>test</argument>
        </command>
      </postDestinationCall>
    </transferSet>
  </managedTransfer>
</request>
```

Related concepts

[“Specifying programs to run with MFT” on page 239](#)

You can run programs on a system where a Managed File Transfer Agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference

[“File transfer request message format” on page 755](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

MFT monitor request message formats

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the `fteCreateMonitor` command or using the IBM MQ Explorer interface.

The monitor XML must conform to the `Monitor.xsd` schema using the `<monitor>` element as the root element.

Monitor messages can have one of the following root elements:

- <monitor> - for creating and starting a new resource monitor
- <deleteMonitor> - for stopping and deleting an existing monitor

There is no command message for the fteListMonitors command because the command directly retrieves matching monitor definitions from the SYSTEM.FTE topic.

Schema

The following schema describes which elements are valid in a monitor request XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/
  MonitorDefinition"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">
  <xsd:include schemaLocation="FileTransfer.xsd"/>
  <xsd:element name="monitor">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="monitorNameType"
          minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string"
          minOccurs="0" maxOccurs="1"/>
        <xsd:element name="pollInterval" type="pollIntervalType"
          minOccurs="1" maxOccurs="1"
          default="10"/>
        <xsd:element name="batch" type="batchType"
          minOccurs="0" maxOccurs="1"/>
        <xsd:element name="agent" type="agentNameType"
          minOccurs="1" maxOccurs="1"/>
        <xsd:element name="resources" type="monitorResourcesType"
          minOccurs="0" maxOccurs="1"/>
        <xsd:element name="triggerMatch" type="triggerMatchType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="tasks" type="monitorTasksType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="originator" type="origRequestType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="defaultVariables" type="defaultVariablesType"
          maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="deleteMonitor">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="monitorNameType"
          minOccurs="1" maxOccurs="1"/>
        <xsd:element name="originator" type="origRequestType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType"
          maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="transferRequestType">
    <xsd:choice>
      <xsd:element name="managedTransfer" type="managedTransferType"/>
      <xsd:element name="managedCall" type="managedCallType"/>
    </xsd:choice>
    <xsd:attribute name="version" type="versionType"/>
  </xsd:complexType>
  <xsd:complexType name="monitorResourcesType">
    <xsd:choice>
      <xsd:sequence>
        <xsd:element name="directory" type="monitoredDirectoryType">
```

```

        minOccurs="1"    maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:element name="queue" type="monitoredQueueType"/>
</xsd:choice>
</xsd:complexType>

<xsd:complexType name="monitoredDirectoryType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="recursionLevel"
type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="id" type="resourceIdAttrType"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="monitoredQueueType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="id" type="resourceIdAttrType"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="triggerMatchType">
    <xsd:sequence>
        <xsd:element name="conditions" type="conditionsType"
minOccurs="1"    maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="conditionsType">
    <xsd:choice minOccurs="1">
        <xsd:element name="allOf" type="listPredicateType"
minOccurs="1"    maxOccurs="1"/>
        <xsd:element name="anyOf" type="listPredicateType"
minOccurs="1"    maxOccurs="1"/>
        <xsd:element name="condition" type="conditionType"
minOccurs="1"    maxOccurs="1"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="listPredicateType">
    <xsd:choice>
        <xsd:element name="condition" type="conditionType"
minOccurs="1"    maxOccurs="unbounded"/>
    </xsd:choice>
</xsd:complexType>

<xsd:complexType name="conditionType">
    <xsd:sequence>
        <xsd:element name="name" type="conditionNameType"
minOccurs="0"    maxOccurs="1"/>
        <xsd:element name="resource" type="resourceIdType"
minOccurs="0"    maxOccurs="1"/>
        <xsd:choice minOccurs="1">
            <xsd:element name="fileMatch"
type="fileMatchConditionType"
minOccurs="1"    maxOccurs="1"/>
            <xsd:element name="fileNoMatch"
type="fileNoMatchConditionType"
minOccurs="1"
maxOccurs="1"/>
            <xsd:element name="fileSize"
type="fileSizeConditionType"
minOccurs="1"    maxOccurs="1"/>
            <xsd:element name="queueNotEmpty"
type="queueNotEmptyConditionType"
minOccurs="1"    maxOccurs="1"/>
            <xsd:element name="completeGroups"
type="completeGroupsConditionType"
minOccurs="1"    maxOccurs="1"/>
            <xsd:element name="fileSizeSame"
type="fileSizeSameType"
minOccurs="1"    maxOccurs="1"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
minOccurs="0"    default="*.*"/>
    </xsd:sequence>
</xsd:complexType>

```

```

        <xsd:element name="exclude" type="conditionPatternType"
            minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileNoMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
            minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
            minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileSizeConditionType">
    <xsd:sequence>
        <xsd:element name="compare" type="sizeCompareType"
            minOccurs="1" default="0" />
        <xsd:element name="pattern" type="conditionPatternType"
            minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
            minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="sizeCompareType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="operator" type="sizeOperatorType"
                use="required" />
            <xsd:attribute name="units" type="fileSizeUnitsType"
                use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="sizeOperatorType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value=">=" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="fileSizeUnitsType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[bB] | [kK][bB] | [mM][bB] | [gG][bB]" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionPatternType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="type" type="patternTypeAttributeType"
                use="optional" default="wildcard" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="patternTypeAttributeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="regex" />
        <xsd:enumeration value="wildcard" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionNameType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string" />
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="queueNotEmptyConditionType" />
<xsd:complexType name="completeGroupsConditionType" />

<xsd:complexType name="fileSizeSameType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
            minOccurs="1" maxOccurs="1" />
        <xsd:element name="exclude" type="conditionPatternType"
            minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="polls" type="positiveIntegerType" use="required" />

```

```

</xsd:complexType>

<xsd:complexType name="pollIntervalType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:int">
      <xsd:attribute name="units" type="timeUnitsType"
        use="optional" default="minutes"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="batchType">
  <xsd:attribute name="maxSize" type="positiveIntegerType" use="required"/>
</xsd:complexType>

<xsd:simpleType name="timeUnitsType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="seconds"/>
    <xsd:enumeration value="minutes"/>
    <xsd:enumeration value="hours"/>
    <xsd:enumeration value="days"/>
    <xsd:enumeration value="weeks"/>
    <xsd:enumeration value="months"/>
    <xsd:enumeration value="years"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="monitorTasksType">
  <xsd:sequence>
    <xsd:element name="task" type="monitorTaskType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorTaskType">
  <xsd:sequence>
    <xsd:element name="name" type="monitorTaskNameType"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="description" type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="transfer" type="transferTaskType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="transferTaskType">
  <xsd:sequence>
    <xsd:element name="request" type="transferRequestType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="resourceIdType">
  <xsd:attribute name="id" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="resourceIdAttrType">
  <xsd:restriction base="xsd:string"></xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="monitorNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="^[^%]*"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="agentNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[%_0-9A-Z]*"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="monitorTaskNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value=".*"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="defaultVariablesType">
  <xsd:sequence>
    <xsd:element name="variable" type="variableType"
      maxOccurs="unbounded" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

```

```

    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="variableType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="key" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>

```

Understanding the create monitor message

The elements and attributes used in create monitor messages are described:

Element descriptions

<monitor>

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<name>

The name of the monitor, unique within the monitor's agent.

<description>

Description of the monitor (not currently used).

<pollInterval>

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: <ul style="list-style-type: none"> • seconds • minutes • hours • days • weeks • months • years

<agent>

Name of the agent the monitor is associated with.

<resources>

Group element that contains the elements specifying the resources to monitor.

<directory>

Fully qualified path specifying the directory on the monitor's agent machine to monitor.

Attribute	Description
recursionLevel	The number of subdirectories to monitor in addition to the specified directory.
id	Unique identifier for the resource.

<queue>

Queue name specifying the queue to monitor on the monitoring agent's queue manager.

<triggerMatch>

Group element that contains the elements specifying the trigger conditions to compare with the monitored resource.

<conditions>

Group element that contains the elements specifying the type of condition to compare with the monitored resource.

<allOf>

Predicate that specifies that all contained conditions must be satisfied.

<anyOf>

Predicate that specifies that any contained conditions must be satisfied.

<condition>

Defines a comparison condition that will contribute to the overall monitor trigger condition.

<name>

Name of the condition.

<resource>

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:

- queueNotEmpty
- completeGroups

<fileMatch>

Group element for a file name match condition.

<pattern>

Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

<fileNoMatch>

Group element for an inverse file name match condition.

<pattern>

Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

<fileSize>

Group element for a file size comparison.

<compare>

Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only '>=' is supported.
units	Specifies file size units, which can be one of: <ul style="list-style-type: none"> • B - bytes

Attribute	Description
	<ul style="list-style-type: none"> • KB - kilobytes • MB - megabytes • GB - gigabytes <p>The units value is case insensitive, so mb' works as well as MB'.</p>

<pattern>

File name pattern to match. Default is * (any file will match).

<queueNotEmpty>

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

<completeGroups>

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

<reply>

Optional element that is used to specify reply queue for asynchronous requests.

Attribute	Description
QMGR	Queue manager name.

<tasks>

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

<task>

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

<name>

Name of the task. Accepts any alphanumeric characters.

<description>

Description of the task. Any text value is allowed.

<transfer>

Group element that defines a transfer task.

<request>

Group element that defines the type of task. This must contain one of the following elements which are inherited from the FileTransfer.xsd schema definition:

- [managedTransfer](#)
- managedCall

Attribute	Description
version	Version of the request as provided by Managed File Transfer. This is in the form n.mm where n is the major release version and mm is the minor version. For example 1.00.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<job>

Group element containing job information.

<jobName>

Specifies logical job identifier.

<defaultVariables>

Group element containing one or more variable elements. These variables are used in variable substitution when monitoring a queue. For more information about variable substitution, see [“Customizing MFT resource monitor tasks with variable substitution”](#) on page 172.

<variable>

Element containing the value associated with the key given by the key attribute.

Attribute	Description
key	The name of the default variable.

Understanding the delete monitor message

The elements and attributes used in delete monitor messages are described:

Element descriptions**<deleteMonitor>**

Group element containing all the elements required to stop and delete a monitor.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

<name>

Name of monitor to delete.

<originator>

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

<reply>

Specifies the name of the temporary reply queue generated for the request. The name of the queue is as defined by the key `dynamicQueuePrefix` in the `command.properties` configuration file. If this is not specified, the queue name has a default value of `WMQFTE`.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor requests:

- [Create a monitor](#)

- Delete a monitor

MFT monitor request message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

Create monitor request

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
  <pollInterval>1</pollInterval>
  <agent>US2.BINDINGS.FILE</agent>
  <resources>
    <directory recursionLevel="0">/srv/nfs/incoming</directory>
  </resources>
  <triggerMatch>
    <conditions>
      <allOf>
        <condition>
          <fileMatch>
            <pattern>*.completed</pattern>
          </fileMatch>
        </condition>
      </allOf>
    </conditions>
  </triggerMatch>
  <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003702</reply>
  <tasks>
    <task>
      <name/>
      <transfer>
        <request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          version="4.00"
          xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
          <managedTransfer>
            <originator>
              <hostName>example.com.</hostName>
              <userID>mqm</userID>
            </originator>
            <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
            <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
            <transferSet>
              <item checksumMethod="MD5" mode="binary">
                <source disposition="leave" recursive="false">
                  <file>/srv/nfs/incoming/*.txt</file>
                </source>
                <destination exist="error" type="directory">
                  <file>/srv/backup</file>
                </destination>
              </item>
            </transferSet>
          </managedTransfer>
        </request>
      </transfer>
    </task>
  </tasks>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
</monitor:monitor>
```

Delete monitor request

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:deleteMonitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
```

```

<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
</originator>
<reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003705</reply>
</monitor:deleteMonitor>

```

Related reference

[“MFT monitor request message formats” on page 772](#)

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the `fteCreateMonitor` command or using the IBM MQ Explorer interface.

Ping MFT agent request message format

You can ping an agent by issuing an **ftePingAgent** command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the `PingAgent.xsd` schema. After you have installed Managed File Transfer, you can find the `PingAgent.xsd` schema file in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/schema`. The `PingAgent.xsd` schema imports `fteutils.xsd`, which is in the same directory.

When the agent receives a ping agent request message on its command queue, if the agent is active, it returns an XML response message to the command or application that put the ping agent request message on the command queue. The response message from the agent is in the format defined by `Reply.xsd`. For more information about this format, see [“MFT agent reply message format” on page 782](#).

Schema

The following schema describes which elements are valid in an ping agent request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent">

  <xsd:include schemaLocation="fteutils.xsd"/>

  <xsd:element name="pingAgent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>

```

Understanding the ping agent request message

The elements and attributes used in the ping agent request messages are described in the following list:

<pingAgent>

Group element containing all the elements required to specify a ping agent request.

<originator>

Group element containing all the elements required to specify the originator of the ping request.

<hostName>

The host name of the machine where the request originated.

<userID>

The user name of the originator of the request.

<mqmdUserID>

The MQMD user name of the originator of the request.

<agent>

The agent to ping.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The queue manager that the agent connects to.

<reply>

The name of the queue for the agent to send the reply message to.

Attribute	Description
QMGR	Required. The name of the queue manager where the reply queue is located.

Example

This example shows a ping agent message sent to the agent AGENT_JUPITER. If AGENT_JUPITER is active and able to process agent requests, it sends a response message to the queue WMQFTE.4D400F8B20003708 on QM_JUPITER.

```
<?xml version="1.0" encoding="UTF-8"?>
<ping:pingAgent xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:ping="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  version="4.00">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <agent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003708</reply>
</ping:pingAgent>
```

MFT agent reply message format

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the Reply.xsd schema. The Reply.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The Reply.xsd schema imports `fteutils.xsd`, which is in the same directory.

Schema

The following schema describes which elements are valid in a reply XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="TransferLog.xsd"/>
  <xsd:element name="reply">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="transferSet" type="transferSetType" minOccurs="0"
maxOccurs="1"/>
        <xsd:element name="status" type="statusType" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
      <xsd:attribute name="detailedReplyMessagesDisabled" type="xsd:boolean"
use="optional"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Understanding the reply message

The elements and attributes used in the reply messages are described in the following list:

<reply>

Element containing the elements that specify the reply information.

Attribute	Description
ID	The ID of the reply.
version	The version of the reply message format.
detailedReplyMessagesDisabled	A notification that the agent has disabled the detailed reply enableDetailedReplyMessages agent property is set to false).

<transferSet>

Specifies the transfer result information of the files requested for transfer. For more information, see [“File transfer log message formats” on page 723](#).

<status>

The status of the action that the agent was requested to perform.

Attribute	Description
resultCode	The result code returned from the action that the agent performed.

<supplement>

Additional response information about the action that the agent was requested to perform.

Example

In the following section is an example reply message:

```
<reply version="1.00"
      xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="Reply.xsd"
      ID="0102020300000000000000000000000000000000000000000000000000000000">
  <status resultCode="65">
    <supplement>Additional reply information</supplement>
  </status>
</reply>
```

MFT diagnostic messages

Diagnostic messages are available here in numerical order, grouped according to the part of Managed File Transfer from which they originate.

For details of these messages, see IBM Documentation: https://www.ibm.com/docs/SSFKSJ_9.0.0/com.ibm.wmqfte.doc/messages_main.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com[®], are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (<http://www.eclipse.org/>).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.



Part Number:

(1P) P/N: