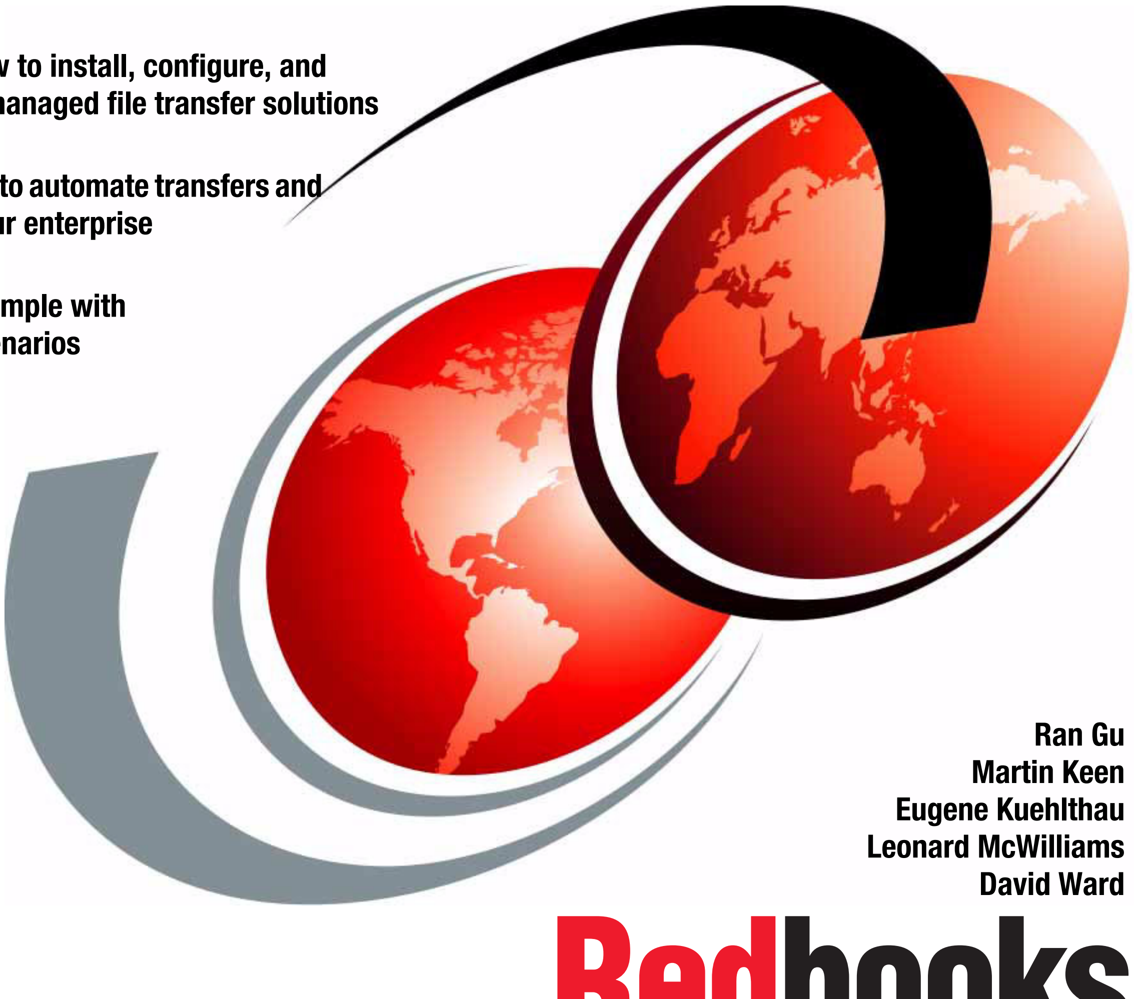
WebSphere® software

IBM

Getting Started with WebSphere MQ File Transfer Edition V7



Discover how to install, configure, and administer managed file transfer solutions

Identify how to automate transfers and integrate your enterprise

Learn by example with practical scenarios

Ran Gu
Martin Keen
Eugene Kuehlthau
Leonard McWilliams
David Ward

Redbooks

ibm.com/redbooks

International Technical Support Organization

Getting Started with WebSphere MQ File Transfer Edition V7

August 2009

SG24-7760-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

First Edition (August 2009)

This edition applies to WebSphere MQ File Transfer Edition V7.0.1

Contents

Notices

Trademarks

Preface

WebSphere MQ File Transfer Edition provides an enterprise-ready managed file transfer capability that is both robust and easy-to-use. WebSphere MQ File Transfer Edition exploits the proven reliability and connectivity of WebSphere MQ to transfer files across a wide range of platforms and networks.

In this IBM® Redbooks® publication, we provide a complete guide for getting started with WebSphere MQ File Transfer Edition. In Part one of the book, we provide a technical overview of the product and provide installation, configuration, and administration guidance for distributed and z/OS® platforms.

In Part two, we provide a series of scenarios to show how you can use WebSphere MQ File Transfer Edition to create managed file transfer solutions. These scenarios range from simple point-to-point transfers through to resource monitoring and complex scripted transfers that include the use of Apache Ant. We describe each scenario step-by-step, which allows you to follow along in your own environment.

In Part three of this book, we discuss topics, such as security, user exits, and the integration of WebSphere MQ File Transfer Edition with other products and solutions.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

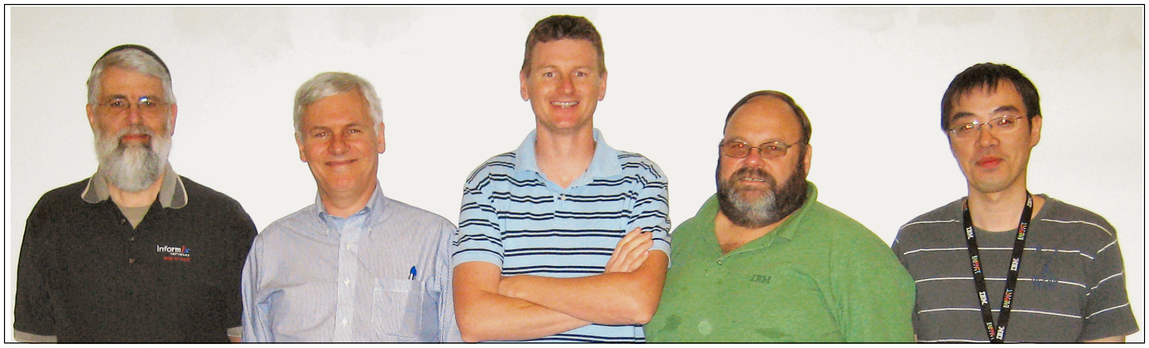


Figure 1 The team (from left-to-right): Leonard, David, Martin, Eugene, and Ran Gu

Ran Gu has a Bachelors degree of Computer Science and Technology from TsingHua University, China. After Ran graduated from the Chinese Academy of Sciences in 2004 with a Masters degree in Software Engineering, he joined IBM China as an IT Specialist, working for five years in WebSphere® technical sales support. Ran's responsibility is to help customers build ESB and data transfer solutions based on IBM products of WebSphere MQ/File Transfer Edition, WebSphere Message Broker, WebSphere Partner Gateway, and WebSphere Transformation Extender.

Martin Keen is a Consulting IT Specialist at the ITSO, Raleigh Center. He writes extensively about WebSphere products and service-oriented architecture (SOA). He also teaches IBM classes worldwide about WebSphere, SOA, and enterprise service bus (ESB). Before joining the ITSO, Martin worked in the EMEA WebSphere Lab Services team in Hursley, United Kingdom (UK). Martin has a bachelors degree in Computer Studies from the Southampton Institute of Higher Education.

Eugene Kuehlthau is a Software IT Specialist in the USA. He has 35 years of experience in the field of data processing. His areas of expertise include WebSphere MQ, WebSphere MQ Workflow, and WebSphere Process Server. He has also developed WebSphere MQ and Workflow courses.

Leonard McWilliams is a Consulting IT Specialist on the WebSphere Federal sales team working primarily with WebSphere MQ and WebSphere Message Broker in classified accounts. He has 35 years of IT industry consulting and application development experience, which includes messaging, database management, and geographical information systems. He has a Bachelors degree in Biological Science, Philosophy, and Music from the University of Kansas and a Masters in Education from Antioch University.

David Ward is Senior Software Engineer with the IBM Software Group in the United States. He has 30 years of experience in software architecture and design and network and systems development. His areas of expertise include enterprise messaging and software development methodologies.

Thanks to the following people for their contributions to this project:

Ben Mann
IBM WebSphere MQ File Transfer Edition Product Manager

Richard Cumbers, Adrian Preston, Peter Verdon, Rob Breeds, Karen Rodgers, Katherine Shann, Christopher Harris, Geoff Judd
IBM WebSphere MQ File Transfer Edition development

Humayun Bhyat
IBM WebSphere MQ FIle Transfer Edition Client Technical Professional

Adam McWilliams
Chief of IT Engineering and Strategy, US Peace Corps

Rich Conway
IBM ITSO

T.Rob Wyatt
IBM WebSphere MQ Security Focused Practice

Robert Simons
IBM Senior Managing Consultant, WebSphere MQ File Transfer Edition Champion

Patrick T. Verdugo
IBM Technical Sales Manager – Connectivity, Worldwide WebSphere Technical Sales

Clyde Bonnema
IBM Worldwide Connectivity Technical Sales Support

Jackie Swett
IBM WebSphere IT Specialist, Federal Software Sales

Robert G. Herbison
IBM Federal WebSphere Technical Sales, Connectivity & Business Integration

Carl Farkas
IBM SW IOT TechWorks zWebSphere Application Integration consultant

Bhargav Perepa
IBM WebSphere IT Specialist, Federal Software Sales

Stefano Marinoni
IBM Advisory IT Specialist, WebSphere Technical Sales

Luca Amato
IBM Senior IT Architect, WebSphere Technical Sales

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Part 1

WebSphere MQ File Transfer Edition

1

Overview of WebSphere MQ File Transfer Edition

In this chapter, we provide an overview of the IBM WebSphere MQ File Transfer Edition product (FTE). WebSphere MQ File Transfer Edition is part of the IBM WebSphere MQ family and provides a complete managed file transfer solution that leverages existing MQ networks and skills.

The topics that we discuss in this chapter are:

1.1 What is managed file transfer?

The term *managed file transfer* encompasses the need that organizations have to configure, track, and audit file transfer activity in a consistent way. The requirements for managed file transfer are:

- Auditability

  File transfer activity must be logged so that administrators can determine where each file is sent and when the transfer occurred. The transfer log must be centrally accessible.

- Security

  File transfer requests should only be accepted from authorized people or application systems.

- Recoverability and reliability

  Network or other errors that might interrupt a transfer should not cause the transfer to be abandoned or partial files to be received.

- Platform connectivity

  File transfers must span multiple platforms.

1.2 What is WebSphere MQ File Transfer Edition?

WebSphere MQ File Transfer Edition provides an enterprise-ready managed file transfer capability that is both robust and easy-to-use. WebSphere MQ File Transfer Edition exploits the proven reliability and connectivity of WebSphere MQ to transfer files across a wide range of platforms and networks. In addition to leveraging existing WebSphere MQ networks, you can easily integrate WebSphere MQ File Transfer Edition with existing file transfer systems. Figure 1-1 shows the FTE and MQ family.

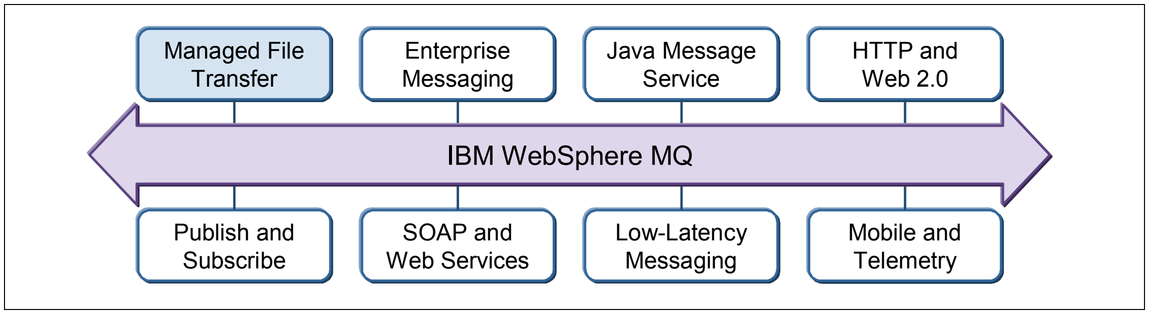


Figure 1-1 WebSphere MQ File Transfer Edition and the MQ family

The benefits that WebSphere MQ File Transfer Edition offers are:

- Auditability

  WebSphere MQ File Transfer Edition provides full logging of transfers at both the source and destination systems. File transfer audit logs are stored in WebSphere MQ queues and optionally in a relational SQL database.

- Ease-of-use

  Using WebSphere MQ File Transfer Edition, you can initiate file transfers using the graphical user interface in WebSphere MQ Explorer, command-line commands, and scripts.

- Simplicity

  WebSphere MQ File Transfer Edition has a low resource footprint and, apart from WebSphere MQ, has no other pre-requisite software.

- Security

  Access to files is controlled by file-system permissions. File transfers can be protected using SSL encryption and authentication.

- Automation

  File transfers can be set up to occur at specified times or dates or repeated at specified intervals. File transfers can also be triggered by a range of system events, such as new files or updated files.

1.3 Architecture of WebSphere MQ File Transfer Edition

WebSphere MQ File Transfer Edition is comprised of a number of components that are all supported by one or more WebSphere MQ queue managers in the network. These components are:

- Agents

  FTE agents are programs that perform the fundamental file transfer function, for example they send and receive files from the local system.

- Configuration commands

  Commands that are used to control FTE from a command-line. Configuration commands perform tasks, such as creating and deleting agents.

- Administration commands

  Administration commands perform tasks, such as creating new file transfers.

- Graphical User Interface

  Provides a point-and-click graphical interface to configure and administer FTE.
- Database Logger

  Sends the contents of file transfer log messages to a database.

The components of WebSphere MQ File Transfer Edition use WebSphere MQ to communicate with each other, and the agents in particular use WebSphere MQ to transport the contents of files across the network to other agents, as shown in Figure 1-2.

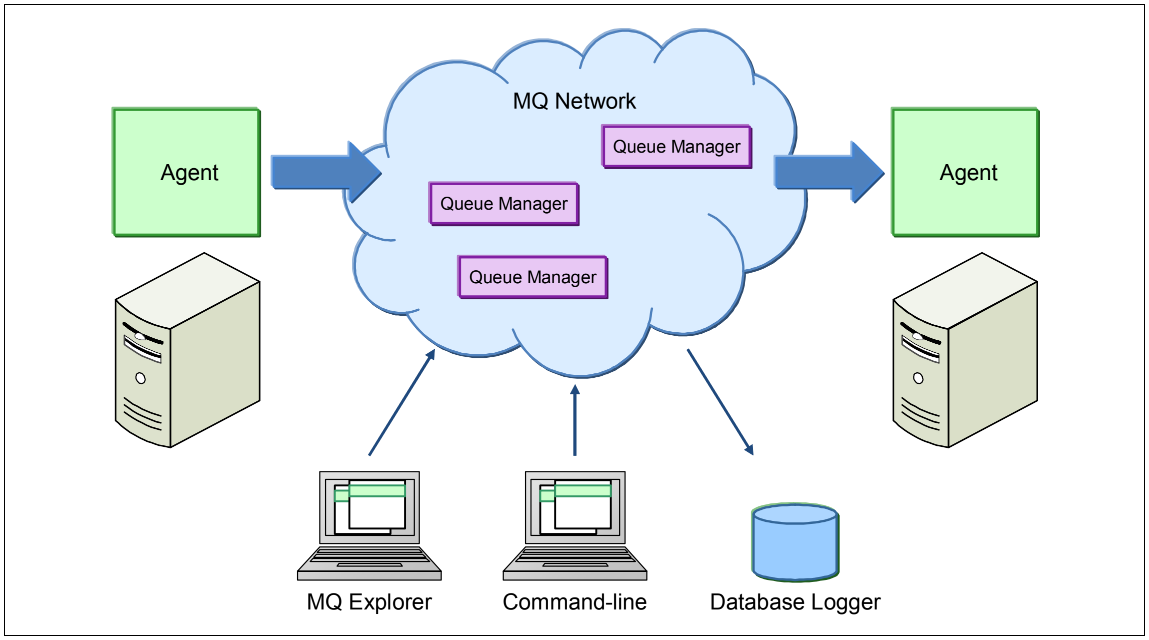


Figure 1-2 Overview of WebSphere MQ File Transfer Edition Architecture

1.3.1 FTE Agents

Agents are FTE programs that form the endpoints for file transfer operations. Essentially, agents perform the fundamental task of transferring files across the network using WebSphere MQ network as the back bone. When requested to send a file, an agent reads the file's contents and sends it to the destination agent in the form of MQ messages. Often, those messages are carried by a WebSphere MQ *channel* across the network where another agent receives them. The receiving agent re-assembles the file on the destination system. There must be an FTE agent running on each host system that can transfer files to or from other systems.

A single agent can process more than one file transfer concurrently, and concurrent transfers might be to the same or different destination agents.

Agents use the WebSphere MQ network to send file information, so every agent needs a queue manager, which is called an *agent queue manager.* An agent queue manager can host more than one agent because each agent uses its own queues, which are separate from the queues that other agents use.

There are two types of FTE agents that correspond to the IBM WebSphere MQ File Transfer Edition *Server* product and the IBM WebSphere MQ File Transfer Edition *Client* product:

- Server Edition agent

  The agent supplied with the FTE Server edition product can connect to a local queue manager using an *MQ bindings connection*. These agents can also connect to a local or remote queue manager using an *MQ client connection.*

- Client Edition agent

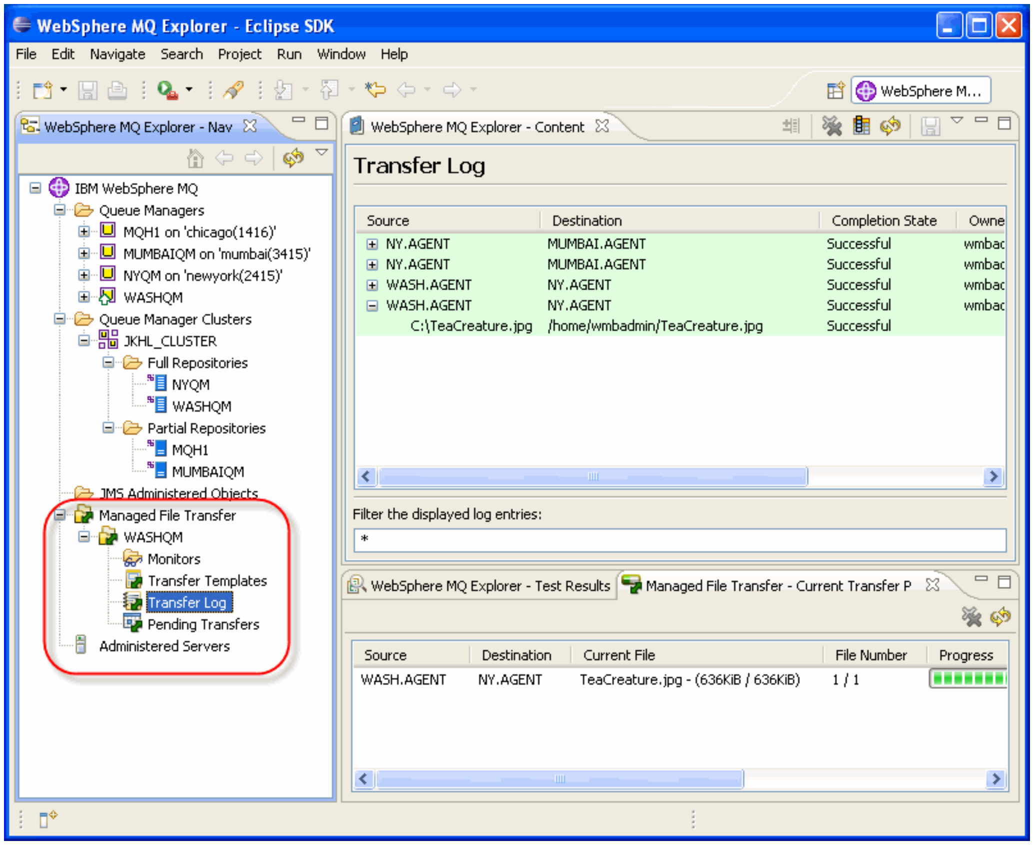
  The agent that is supplied with the FTE Client edition product uses an *MQ client connection* to connect to a queue manager. Client agents can be located on the same system or on a different system than their agent queue manager is located on.

1.3.2 Graphical user interface

You can administer WebSphere MQ File Transfer Edition with the MQ Explorer workbench, using the FTE GUI plug-in. The FTE GUI plug-in is part of the IBM WebSphere MQ File Transfer Edition Remote Tools and Documentation product.

WebSphere MQ Explorer is available for Windows® and Linux® platforms, is supplied with WebSphere MQ, and in standalone form with WebSphere MQ MS0T SupportPac.

Figure 1-3 on page 8 shows the WebSphere MQ Explorer views for managing WebSphere MQ File Transfer Edition.



Source	Destination	Completion State	Owne
NY.AGENT	MUMBAI.AGENT	Successful	wmbac
NY.AGENT	MUMBAI.AGENT	Successful	wmbac
WASH.AGENT	NY.AGENT	Successful	wmbac
WASH.AGENT	NY.AGENT	Successful	wmbac
C:\TeaCreature.jpg	/home/wmbadmin/TeaCreature.jpg	Successful	

Source	Destination	Current File	File Number	Progress
WASH.AGENT	NY.AGENT	TeaCreature.jpg - (636KiB / 636KiB)	1 / 1	

Figure 1-3 Administering WebSphere MQ File Transfer Edition using WebSphere MQ Explorer

1.3.3 Command-line tools

Use the command-line tools to configure WebSphere MQ File Transfer Edition and to operate it (for example, submit and monitor file transfer requests).

The configuration commands are used to set up in FTE and are needed when you create and configure a new FTE installation. The commands are:

- fteCreateAgent
- fteDeleteAgent
- fteChangeDefaultConfigurationOptions
- fteSetupCoordination
- fteSetupCommands

The administration commands in the next bulleted list are used to *operate* FTE and supports tasks that are typically performed day-to-day in an FTE installation:

- fteStartAgent
- fteCreateTransfer
- fteCreateMonitor
- fteDeleteMonitor
- fteListAgents
- fteShowAgentDetails
- fteListScheduledTransfers
- fteDeleteScheduledTransfer
- fteCleanAgent
- fteStopAgent
- fteCancelTransfer
- fteSetAgentTraceLevel
- fteListMonitors
- fteAnt
- ftePingAgent
- fteStartDatabaseLogger
- fteStopDatabaseLogger

1.3.4 Queue managers

WebSphere MQ File Transfer Edition uses WebSphere MQ for communicating between its agents, the WebSphere MQ Explorer plug-in, and the command-line commands. Further, FTE uses WebSphere MQ to transmit bulk file data between agents.

To do its job, each component of FTE needs to connect to a WebSphere MQ queue manager, of which there are three roles:

- Coordination queue manager
- Agent queue managers
- Command queue managers

FTE does not require that all three roles be physically separate queue managers, although there are usually good reasons to design your installation that way. A simple FTE installation can designate a single queue manager to fill all three roles, but doing this in a production environment is not ideal from a performance and reliability point-of-view.

Production environments are best designed using separate coordination and agent queue managers.

The Coordination Queue Manager

The coordination queue manager acts as a central collection point where information about FTE activity is gathered. An FTE network typically has a separate designated queue manager to be the coordination queue manager. Agents publish active file transfer status information to a topic that is hosted on this queue manager. Additionally, the coordination queue manager broadcasts file transfer audit information to other FTE components and to any interested and authorized parties who subscribed to FTE information topics.

The coordination queue manager's primary role is to collect information about the FTE network, and unless the coordination queue manager is also hosting WebSphere MQ File Transfer Edition agents, does not participate in the transmission of file data (the agent queue managers perform that duty). Of course, it is possible to define a single queue manager that fills both the coordination queue manager role and the agent queue manager role and in that case, the coordination queue manager also carries file data.

FTE requires that the coordination queue manager be hosted using a WebSphere MQ V7 or later installation. Additionally, the coordination queue manager must be enabled for WebSphere MQ publish / subscribe.

The Agent Queue Manager

Each agent connects to its *agent queue manager* and through it receives file transfer requests and publishes its own file transfer start and stop status events to the coordination queue manager.

An agent queue manager hosts the queues that are used by the agents that it supports. Each agent uses its own uniquely-named set of queues so that an agent queue manager can support one or more *server agents* on its local system in addition to one or more *client agents* on remote systems.

The Command Queue Manager

The command-line tools and the FTE MQ Explorer GUI plug-in use the command queue managers to communicate with agents.

1.4 Integration

In this section, we discuss how WebSphere MQ File Transfer Edition can integrate with other components using Apache Ant and pre- and post-processing tasks.

1.4.1 Using Apache Ant

Ant is an XML-based scripting tool, which the Apache Software Foundation releases, that is widely used for building Java™-based software suites. Although its original purpose was to manage building Java software, Ant is becoming popular as a general-purpose scripting tool. WebSphere MQ File Transfer Edition can integrate its file transfer functions using scripts that are run by Ant.

Ant accepts a script file that is coded in XML. Within the XML script are verbs, known as Ant *tasks*, that represent the actions that the script will perform. Ant itself provides many hundreds of tasks to address a wide range of scripting needs.

WebSphere MQ File Transfer Edition provides its own set of Ant tasks that can be used to integrate FTE file transfer processing within an Ant script. The FTE tasks can be combined with any of the other Ant tasks to address more complex file management needs.

WebSphere MQ File Transfer Edition provides the following Ant tasks:

- awaitoutcome
- call
- cancel
- filecopy
- filemove
- ignoreoutcome
- ping
- uuid

A number of the Ant tasks that WebSphere MQ File Transfer Edition provides use nested XML elements to further qualify the operations.

1.4.2 Using file transfer pre and post-processing tasks

When you configure WebSphere MQ File Transfer Edition to send and receive files, it is possible to have FTE run a task both before and after the transfer occurs. Pre-processing tasks are executed before the file transfer, and post-processing tasks are executed after the transfer.

Additionally, you can configure pre and post-processing tasks for either or both the source agent and the destination agent.

Summary

In this chapter, the aspects of WebSphere MQ File Transfer Edition that we discussed were:

- Managed File Transfer concepts
- Introduction to WebSphere MQ File Transfer Edition
- WebSphere MQ File Transfer Edition Architecture
- Integration features of WebSphere MQ File Transfer Edition

2 Topologies

When designing a file transfer topology for WebSphere MQ File Transfer Edition, you can choose to build a topology that supports everything the product has to offer, or you can build a smaller topology that supports just the simple file transfer clients and qualities of service that you need.

In this chapter, we introduce two main production topologies for WebSphere MQ File Transfer Edition. Each topology can be extended with varying usage patterns for system environments, network conditions, and qualities of service. You can use this chapter to learn about these topologies and to determine which production topology is right for you.

The topics that we discuss in this chapter are:

- 2.1, “Single queue manager topology” on page 14
- 2.2, “Multi queue manager topology” on page 20

2.1 Single queue manager topology

In this section, we describe a single queue manager topology and the architectural details for WebSphere MQ File Transfer Edition.

2.1.1 Topology overview

It is possible to have a single queue manager be the WebSphere MQ File Transfer Edition network. This queue manager supports all functions of an WebSphere MQ File Transfer Edition queue manager: coordination queue manager, command queue manager, and agent queue manager.

WebSphere MQ File Transfer Edition Server agents, which run on a machine that has WebSphere MQ installed, can connect to this queue manager in either bindings or client mode and have their own sets of queues on this queue manager. WebSphere MQ File Transfer Edition Client agents, which must run on a machine that does not have WebSphere MQ installed, can only associate with this queue manager in client mode. Either server agents or client agents can send and receive the files that are transferred. The choice of available transport modes depend on the edition of the product that you are using.

Agents register with the coordination queue manager and publish their details to this queue manager. You can use the GUI component of the WebSphere MQ File Transfer Edition plug-in to start transfers from the WebSphere MQ Explorer or to execute FTE commands to trigger and manage file transfers by local or remote tools.

A basic topology on a single queue manager

Figure 2-1 on page 15 shows architectural details of the basic topology on a single queue manager for WebSphere MQ File Transfer Edition.

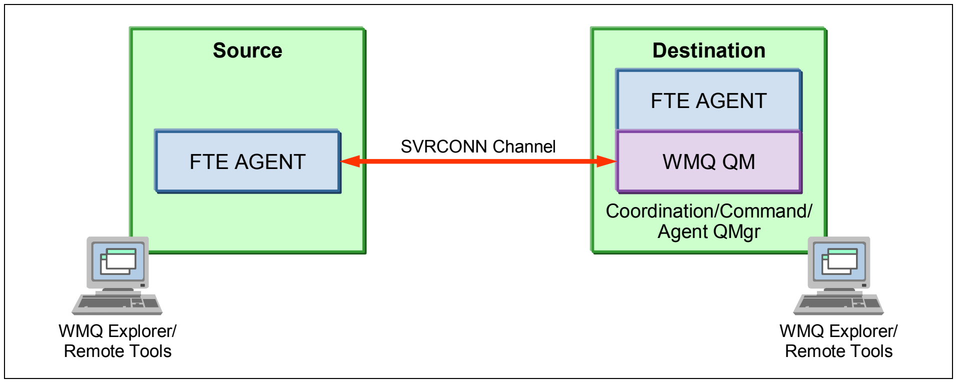


Figure 2-1 A basic topology on single queue manager

The key architectural details of this topology are:

- Coordination queue manager resides on the destination
- Command queue manager and agent queue manager are the same as the Coordination queue manager
- Server agent on the destination connects to the queue manager in bindings mode
- Client agent on the source uses client mode to associate with the only queue manager by a server connection channel
- Either agent can be a source or destination
- Files can be moved by a server connection channel between two agents

This is the simplest type of topology. The source only requires WebSphere MQ File Transfer Edition Client to be installed. WebSphere MQ File Transfer Edition Server is installed and configured with WebSphere MQ V7. In this topology, you can also use WebSphere MQ File Transfer Edition plug-in for WebSphere MQ Explorer and remote tools. Tools on the client agent must be configured to connect to the queue manager by client mode.

An extended topology on a single queue manager

Figure 2-2 on page 16 is an extended topology on a single queue manager with two clients.

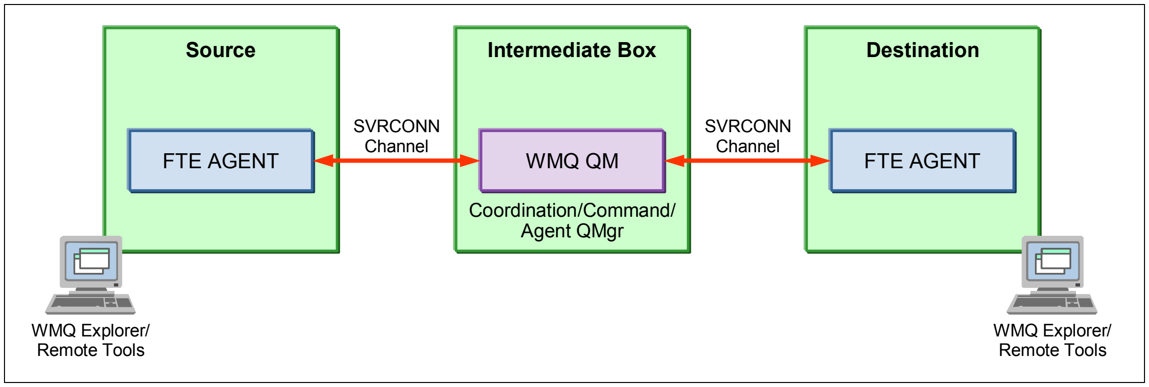


Figure 2-2 An extended topology on a single queue manager

The topology in Figure 2-2 has the following characteristics:

- Coordination queue manager resides on the intermediate queue manager
- The command queue manager and agent queue manager can both reside on the intermediate queue manager
- Each agent uses client mode to associate with the intermediate queue manager through a server connection channel
- Either client agent can be a source or destination
- Two client agents do not directly transfer files with each other but communicate through the intermediate queue manager
- The intermediate queue manager does not have any FTE code installed

In the topology, you can install the WebSphere MQ File Transfer Edition plug-in or remote tools for operating, monitoring, and managing in every FTE client agent.

2.1.2 Benefits and drawbacks of this topology

In this section, we discuss the benefits and drawbacks of a single queue manager topology.

Benefits

The single queue manager topology is a light weight file transfer architecture that is simple to understand and easy to create. It makes significant use of FTE clients to transfer files between a source and destination.

The benefits of this topology are:

- FTE client reduces system requirements

  You can configure many client agents on a single queue manager topology. These client agents require less system resources than server agents that are running on queue managers by bindings mode. Client agents support more desktop operation system versions, such as Windows 2000, Windows XP, Windows Sever 2003, Windows Vista®, and Windows Server® 2008.

- Installation and configuration are simple

  WebSphere MQ File Transfer Edition client does not depend on a WebSphere MQ client, and it provides a small installation footprint.

- Minimal maintenance and management workload

  There is minimal workload of management and maintenance on the FTE client because there are not any queue managers or queues.

Drawbacks

The limitations of this topology are:

- Limitation of communication between FTE client agents

  As shown in Figure 2-2 on page 16, two client agents cannot directly transfer files with each other. The FTE clients must depend on an intermediate queue manager and communicate using server connection channels. We do not recommend using server connection channels across a wide area network (WAN). Therefore this topology is best suited to intra-enterprise communications.

- Performance limitations

  Client channels perform poorly in comparison to sender/receiver channels because client channels use one TCP/IP connection to complete two-direction communication. You do not get the best transfer performance between physical nodes if you use a client channel.

  Additionally, all workload must flow through a single queue manager, potentially creating a bottleneck in performance.

- No flexibility and scalability

  This is a hub structure on a single queue manager. It is difficult for this structure to be extended to a multi-level structure. There is little flexibility or scalability.

- Outage problem on a single queue manager

  If the single queue manager is offline in this topology, all file transfers cease; therefore, this is a single point-of-failure.
- Security limitations

  In production environments, the security level that is required for firewalls is typically high. Therefore incoming and outgoing ports on the firewall are tightly controlled and limited. The sender/receiver channels in WebSphere MQ can support this security requirement, but the server connection channels that FTE client agents use do not support setting a fixed local address port or a range of ports.

2.1.3 When to use this topology

Although the single queue manager topology is not commonplace in production environments, there are scenarios where it makes sense to use it.

One such scenario is that of a travel agency that hopes to construct a reliable, robust, and safe file transfer platform between two local offices, as shown in Figure 2-3. The travel agency has two offices in New York: the headquarters office and financial office. An intranet environment that is in good condition is between the two offices.

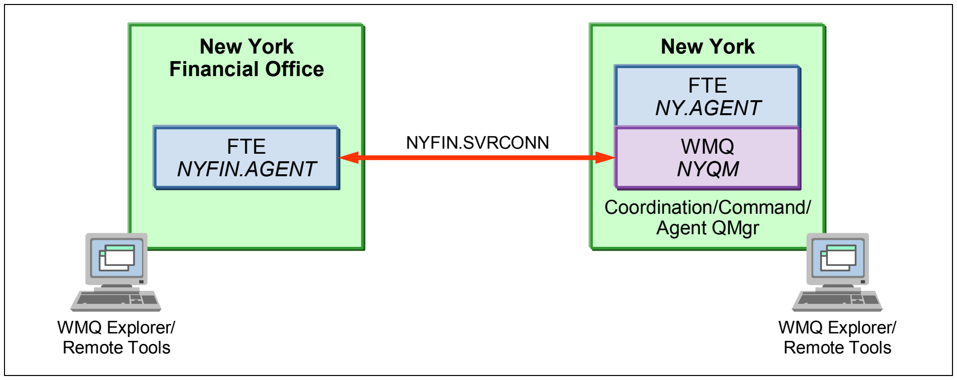

New York Financial Office
FTE NYFIN.AGENT
NYFIN.SVRCONN
New York
FTE NY.AGENT
WMQ NYQM
Coordination/Command/ Agent QMgr
WMQ Explorer/ Remote Tools
WMQ Explorer/ Remote Tools

Figure 2-3 A travel agency scenario

The headquarter office is the business and data center. Every day the headquarter office sends marketing information and travel sales data to the New York financial office. In the same site, there are a few IT engineers, application developers, and system administrators.

The financial office in New York is a local branch office of this travel agency. This office analyzes marketing and sales data every day and then returns the financial analysis reports to the headquarters office. There is no IT engineer for system management and maintenance in the New York office.

The single queue manager topology is a good fit for the travel agency's requirements:

- The headquarters office hosts WebSphere MQ server V7.0 and WebSphere MQ File Transfer Edition server V7.0. A server agent using bindings mode is connected to only one queue manager that shares as coordination queue manager, command queue manager, and agent queue manager.
- In the financial office, a client agent is configured to the queue manager on the headquarter office. Files can be transferred through the NYFIN.SVRCONN channel between the two offices. The two offices can both use the FTE eclipse plug-in tools or command lines to manage and monitor file transfers.

Another scenario is that a public agency adopts a software solution for file distribution and management instead of the high cost hardware solution of a Storage Area Network. The public agency has three business systems and one information center in a local area network. Every day three business systems upload business data files and official documents to the information center. At the same time, the systems download the updated data files from the information center on schedule. There is not a significant amount of file movement.

In Figure 2-4 on page 20, we provide a light-weight software solution to the public agency.

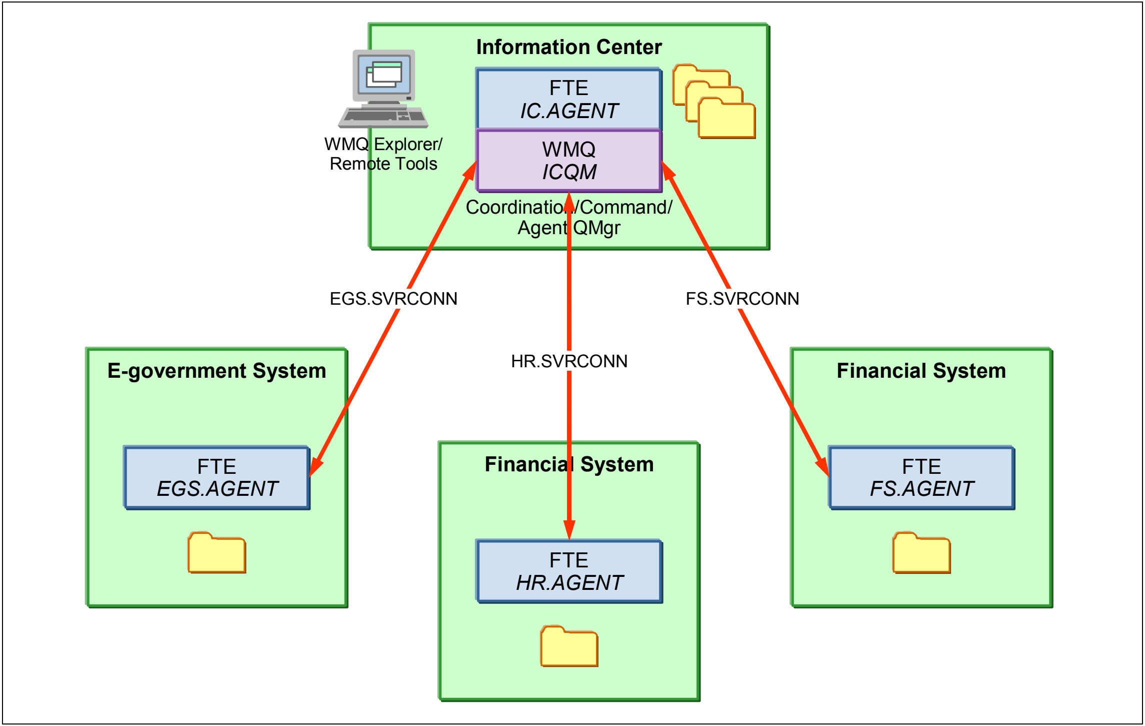


Figure 2-4 A public agency scenario

The information center is a file transfer hub that runs one WebSphere MQ queue manager as a coordination queue manager, command queue manager, and agent queue manager. FTE client agents run on three business systems and connect to the coordination queue manager using the SVRCONN channel in the local area network. This topology can help the public agency implement quick file movement in two-directions and reduce costs.

2.2 Multi queue manager topology

In this section, we introduce a multi queue manager topology and describe the architectural details for WebSphere MQ File Transfer Edition.

2.2.1 Topology overview

The more common WebSphere MQ File Transfer Edition network domain has multiple queue managers that are interconnected. You might have some queue managers that act as WebSphere MQ File Transfer Edition agent concentrators, while other queue mangers might only have binding mode agents. Either server agent or client agent needs a set of queues on an agent queue manager. These queues are internal queue systems to FTE and they are transparent to the end user.

You can have queue managers act as the WebSphere MQ File Transfer Edition command queue manager to provide for entry points for command lines and WebSphere MQ File Transfer Edition plug-in tools. The command queue manager can be different from the agent queue manager. It is not necessary for agents to be connected to the same command queue manager, and this queue manager can be local or remote.

Either command queue manager or agent queue manager can be from WebSphere MQ V6.0 or later.

Finally there is at least a coordination queue manager in a WebSphere MQ File Transfer Edition network domain. The coordination queue manager must be a WebSphere MQ V7.0 queue manager for its publish/subscribe feature. FTE Agents send progress and logging messages to coordination queue manager, then these messages are published to the SYSTEM.FTE topic. The FTE Explorer plug-in, when installed, is a subscriber to this topic.

If there are more than two coordination queue managers in a multi queue manager topology, one agent queue manager must only belong to one coordination queue manager and have connectivity to the queue manager in a WebSphere MQ network.

WebSphere MQ has many ways to connect queue managers, and, by its nature, that means that there are many designs that you can use to develop a multiple queue manager topology for the WebSphere MQ File Transfer Edition network domain.

In the following sections, we introduce some common topologies on multi queue managers.

A basic topology on two queue managers

Figure 2-5 on page 22 shows the architectural details of the basic topology on two queue managers for WebSphere MQ File Transfer Edition.

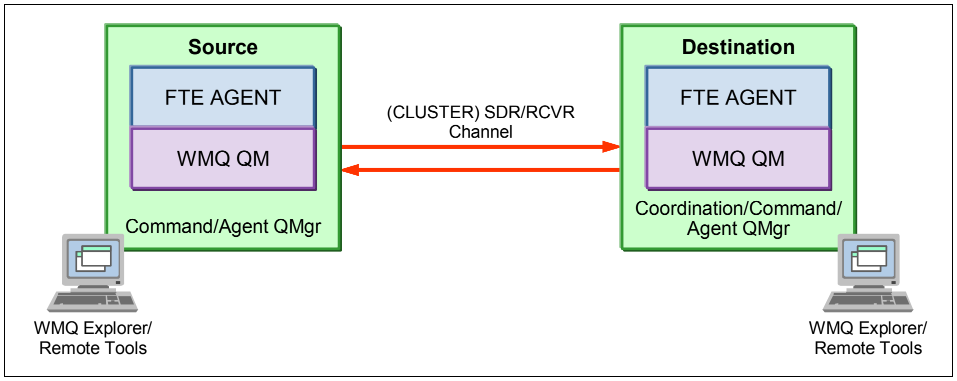

Source
FTE AGENT
WMQ QM
Command/Agent QMgr
(CLUSTER) SDR/RCVR Channel
Destination
FTE AGENT
WMQ QM
Coordination/Command/ Agent QMgr
WMQ Explorer/ Remote Tools
WMQ Explorer/ Remote Tools

Figure 2-5 A basic topology on two queue managers

The key architectural details of the topology are:

- Coordination queue manager can reside on either queue manager
- Command queue manager and agent queue manager can reside on either queue manager
- Each server agent uses bindings mode to connect to its queue manager
- Either server agent can be a source or destination
- Files can be transferred by clustered Sender / Receiver channels between two server agents

Figure 2-5 shows a simple topology with two queue managers. WebSphere MQ File Transfer Edition Server agent is installed on each endpoint and connected to its agent queue manager, which also acts as a command queue manager. You can appoint either queue manager as the coordination queue manager, but this queue manager must be running on WebSphere MQ V7.

In this topology, you can use the WebSphere MQ File Transfer Edition plug-in for WebSphere MQ Explorer and remote tools. However, if the FTE Eclipse plug-in on the source can monitor the transfers, you must the set coordination parameter to connect to the coordination queue manager by client mode.

An extended topology on two queue managers

Figure 2-6 on page 23 is an extended topology on two queue managers.

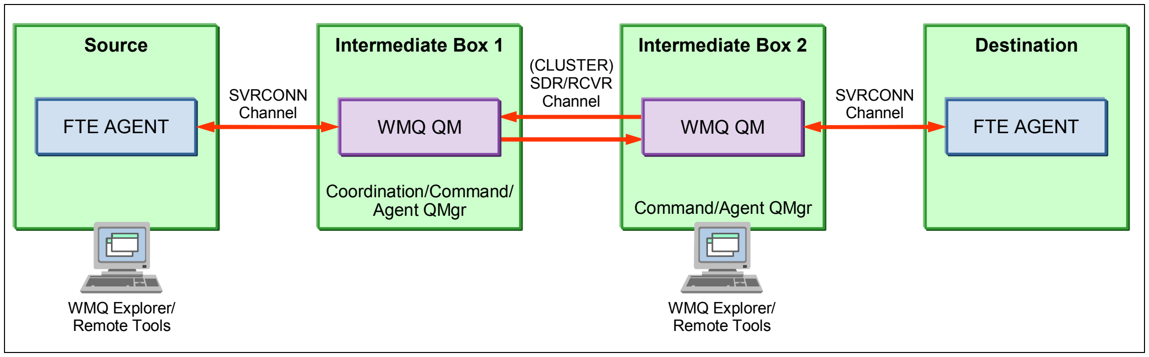


Figure 2-6 An extended topology on two queue managers

The key architectural details of the topology are:

- Coordination queue manager can reside on either intermediate queue manager
- Command queue manager and agent queue manager can reside on either intermediate queue manager
- Each client agent uses client mode to connect to its associated intermediate queue manager by SVRCONN channel
- Either client agent can be a source or destination
- Files can be moved from source to destination by SVRCONN and clustered Sender/Receiver channels
- Intermediate queue managers do not have any FTE code installed

In the topology shown in Figure 2-6, you can actually configure two coordination queue managers, if you want to monitor the transfers on each client agent using the FTE Eclipse plug-in.

A multi-hop topology on multi queue manager

Files are often moved between many nodes, such as three nodes or more. Figure 2-7 on page 24 shows a topology of three queue managers.

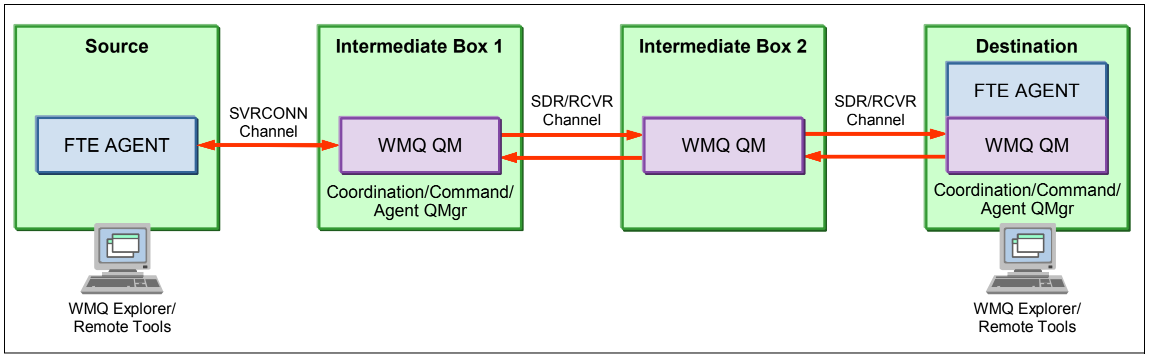


Figure 2-7 A multi-hop topology on multi queue manager

The principal characteristics of this topology are:

- Coordination queue manager can reside on any of the intermediate queue managers or on any other queue manager that is part of the WebSphere MQ network
- Command queue manager and agent queue manager can reside on the queue manager that is near the agent
- Each agent connects to its associated queue manager in client or bindings mode
- Either agent can be a source or destination
- Files can be transferred from source to destination by multi-hop
- The queue manager in the intermediate box does not have any FTE code installed

There is a multi-hop in this topology because there is no direct WebSphere MQ communication link between source and destination. It is possible for files to pass through one or more intermediate queue managers on the way to the destination. For multi-hop, you must define channels between all of the queue managers, transmission queues on the intermediate queue managers, and queue manager aliases on the source and destination. If you want to understand more information about multi-hop or intercommunication between queue managers, refer to *Intercommunication, WebSphere MQ Information Center*:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzae.doc/ic10120_.htm

In Figure 2-7, there are two coordination queue managers. Each agent registers with the closest coordination queue manager and publishes its details to this coordination queue manager. The information about the agents is subscribed to by a command agent on the coordination queue manager. This agent information is subscribed and the WebSphere MQ File Transfer Edition plug-in uses it to allow you to manage and monitor file transfers from the WebSphere MQ Explorer.

It is worth mentioning that every coordination queue manager can only publish progress and logging information of registered agents on it to subscribers, for example, in Figure 2-7 on page 24, the coordination queue manager on Intermediate Box 1 can publish transfer information of the agent on the Source because the agent on Source registered to this closest coordination queue manager. If you configure the FTE Eclipse plug-in tooling on the Source to connect to the coordination queue manager on Intermediate Box 1, you can only receive information from the agent on the Source or other agents that registered to this coordination queue manager.

If you also want to subscribe the transfer and logging information about the other coordination queue manager, you can achieve this through the WebSphere MQ publish/subscribe hierarchy, which is located at Publish/Subscribe User's Guide in the WebSphere MQ Information Center:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.amqnar.doc/ps22740_.htm

A cluster topology on multi queue managers

In this section, we describe a complicated topology on clustered queue managers, as illustrated in Figure 2-8 on page 26.

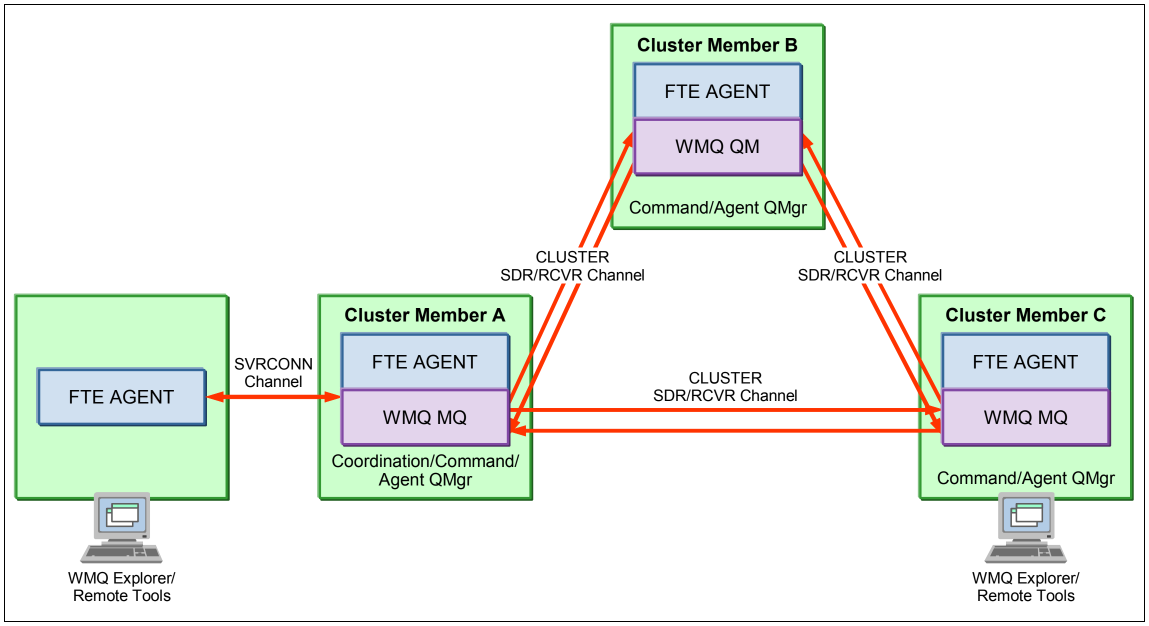


Figure 2-8 A cluster topology on multi queue manager

The characteristics of the topology in Figure 2-8 are:

- Coordination queue manager can reside on either cluster queue manager
- Command queue manager and agent queue manager can reside on either cluster queue manager
- Each agent connects to its associated queue manager in client or bindings mode
- Either agent can be a source or destination
- Files can be moved by Cluster Sender/Receiver channels

In this topology, FTE agents on cluster members can transfer files between each other through cluster sender/receiver channels. Agents outside of the cluster can send files by passing through one or more intermediate queue managers on the way to the destination inside the cluster.

2.2.2 Benefits and drawbacks of the multi queue manager topology

In this section, we discuss the benefits and drawbacks of the multi queue manager topology.

Benefits

Use the multi queue manager topology to transfer high numbers of files. The benefits of this topology are:

- No limitation of network condition

  WebSphere MQ File Transfer Edition server can provide reliable and integrated file transfers. It does not depend on any network condition because FTE server agents run on WebSphere MQ queue managers and adopt clustered Sender/Receiver channels to ensure steady communication. The sender/receiver channels can adapt to the network environment by themselves and reconnect between each other if a channel connection is broken.

- Stable transfer speed and better performance in a WAN

  In a WAN, WebSphere MQ can provide an excellent transfer mechanism and stable performance through clustered Sender/Receiver channels between queue managers. For network bandwidth troubles, you can use multi queue manager topologies to build better performance and stable transfer speed solutions for WebSphere MQ File Transfer Edition.

- Flexibility and scalability

  You can scale the queue manager's architecture horizontally or vertically, for example, you can create a queue manager cluster to simplify configuration and management for file transfers. You can use a queue manager cluster to transfer files among multiple offices, for instance.

  You can also easily add new FTE agents running on queue managers that are connected to the current domain of the WebSphere MQ File Transfer Edition network.

- High-level security

  You can set high-level transport policies on the firewall in the multi queue manager topology, for instance, you can control the firewall two-direction ports and limit a range of ports or fixed port on clustered Sender/Receiver channels. You can also transfer files by multi-hop to pass through the DMZ domain on the way to a destination.

- Outage improvement

  If the queue manager is offline in a single queue manager topology, all agents cannot be used; however, this is not a problem on the multi queue manager topology. Every agent can have an agent queue manager and command queue manager in the WebSphere MQ File Transfer Edition network domain. When an agent queue manager is out-of-service, only agents that run on the offline queue manager are unavailable. Other agents that run on different queue managers are not affected. If the coordination queue manager is independent of command queue managers or agent queue managers, when this coordination queue manager is offline, file transfers between FTE agents continue.

Drawbacks

The drawbacks with the multi queue manager topology are:

- Maintenance of WebSphere MQ objects

  Before you install WebSphere MQ File Transfer Edition, you must first create queue managers, define transmission queues, and configure clustered Sender/Receiver channels. After you complete the FTE installation, you must create a system topic for the coordination queue manager and define system queues for an agent queue manager. Although FTE system objects need not be managed manually, common MQ objects are still maintained for communication between queue managers.

- Cost considerations

  A multi queue manager topology increases the number of software licences above the single queue manager topology. However these licensing costs can be quickly offset by reduced maintenance costs.

2.2.3 When to use the multi queue manager topology

The multi queue manager topology is a common and prevalent solution. The topology on two queue managers is the simplest architecture.

An example of this topology in use is a business bank that plans to build a data load channel between its two data centers. Every day numerous large files above 100 GB are moved in the intranet. The solution ensures that the transfer is reliable and capable of integration between two data centers. In this scenario, we can propose a standard solution and topology of two queue managers, as shown in Figure 2-5 on page 22. Server agents are selected as transfer nodes and provide stable and efficient file movement.

In some scenarios, you will consider a general file transfer platform, which includes three or more nodes as a file transfer solution. These different multi queue manager topologies for customers are recommended for hub, bus, or multi-level structures, for example, an international travel agency has many business processes that use file transfers among their current three offices: Washington, New York, and Mumbai. Washington is the head office of the international travel agency. New York and Mumbai are regional offices.

First, the travel agency implements basic secure file transfers between the head office and regional office, such as financial analysis reports are transferred between Washington and New York and business documents are moved between Washington and Mumbai.

Next, the international travel agency hopes to build a centralized monitoring for file transfer progress and logging into the head office.

The travel agency also wants to solve network problems including transmission stability and efficiency because the network bandwidth between the head office and regional office is limited. Finally, the travel agency will open some new offices for business in the future, and the transfer topology can be extended flexibly.

Figure 2-9 shows this topology for the international travel agency.

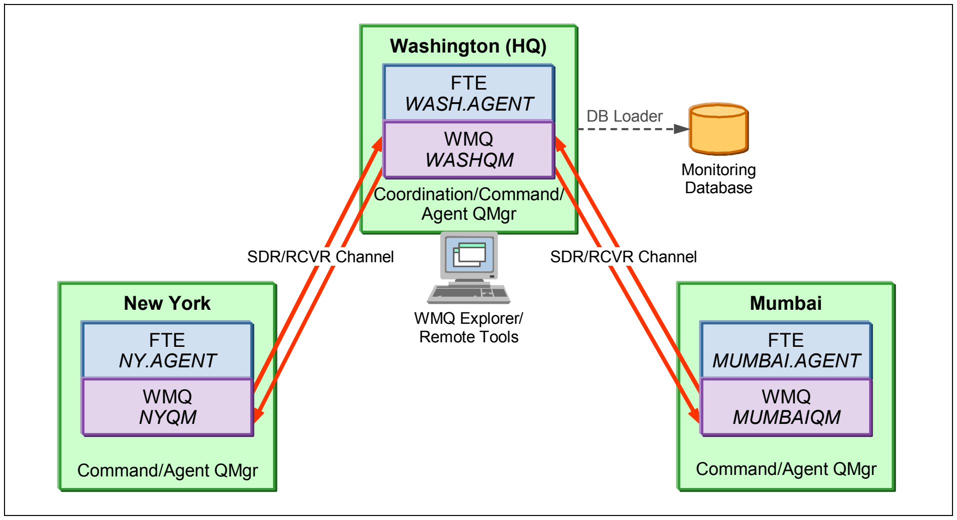


Figure 2-9 File transfer topology of the international travel agency

The key architectural details of the topology for the international travel agency are:

- Coordination queue manager can reside at the head office
- Command queue manager and agent queue manager can reside on either the head office's queue manager or the regional office's queue manager
- Each agent connects to its associated queue manager in bindings mode
- Either agent can be a source or destination
- Files can be transferred by Sender/Receiver channels in the limited intranet

- All transfer logs and messages are concentrated on the coordination queue manager, and the FTE database loading component loads them into the monitoring database
- FTE Eclipse plug-in tools are used to view the current transfer status and history logs
- The topology is easy to extend from regional offices if the international travel agency expands its business range and opens several new offices

Figure 2-10 illustrates a future topology.

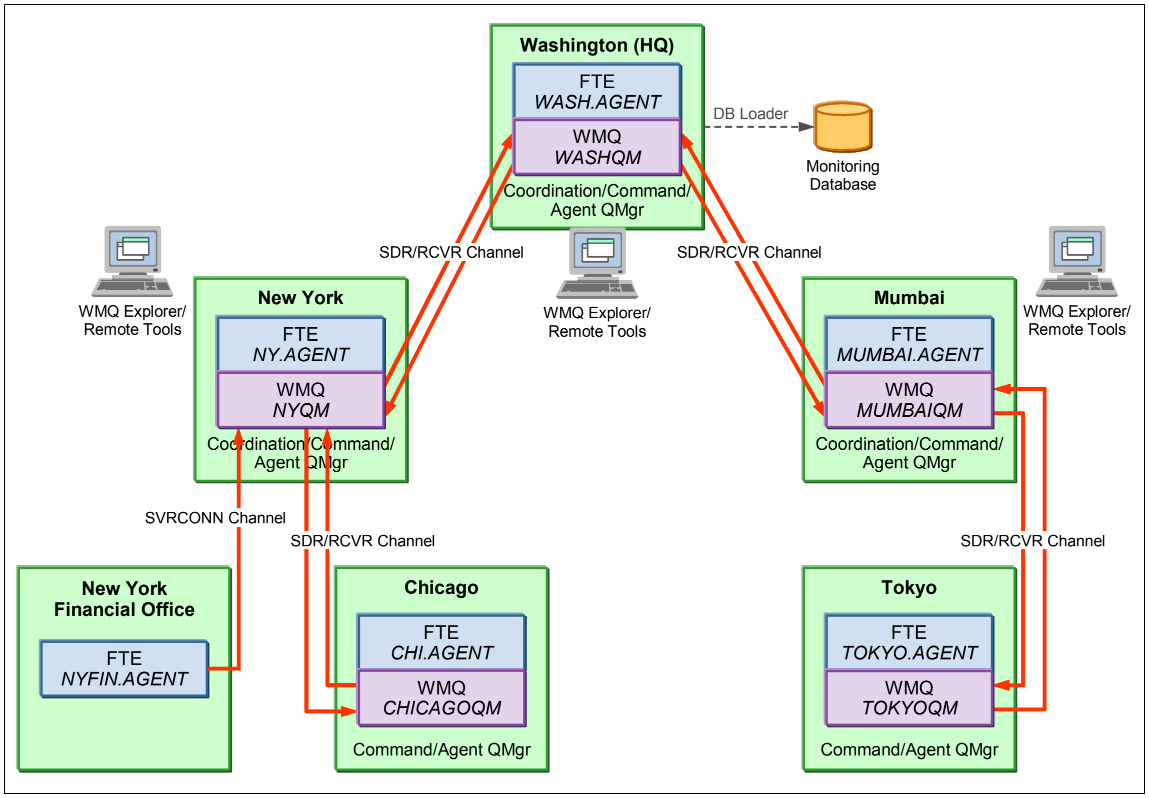


Figure 2-10 The extended topology of the international travel agency in the future

The characteristics of the future topology are:

- A local financial office in New York is added into this topology and connected to the agent queue manager in New York by the SVRCONN channel
- Chicago, the secondary regional office, is included into the WebSphere MQ File Transfer Edition network domain

- The command queue manager and agent queue manager on Chicago are created and connected to the queue manager in New York
- The Tokyo office is also opened as a new office and added into the topology
- Two new coordination queue managers are configured on regional offices: New York and Mumbai
- Every regional office only monitors its own file transfer progress
- The head office monitors the file transfer progress of all of the offices

3

Installing and configuring WebSphere MQ File Transfer Edition

In this chapter, we describe how we installed and configured WebSphere MQ File Transfer Edition (FTE) on Windows and Linux platforms to use with the business scenario that we present in chapters 6 through 9. In Chapter 4, “Configuring WebSphere MQ File Transfer Edition on System z” on page 73, we describe how to install FTE on z/OS.

The environment that we describe in this chapter is used throughout this book as part of the JKHL Travel business scenario, which we describe in Chapter 6, “The business scenario that we use in this book” on page 147.

The topics that we discuss in this chapter are:

3.1 Introduction to the chapter

In this chapter, we describe how we installed and configured WebSphere MQ File Transfer Edition V7.0.1. The topology that we describe is created for a fictional company called JKHL Travel and is used to demonstrate the product features of WebSphere MQ File Transfer Edition throughout this book. Figure 3-1 shows this topology.

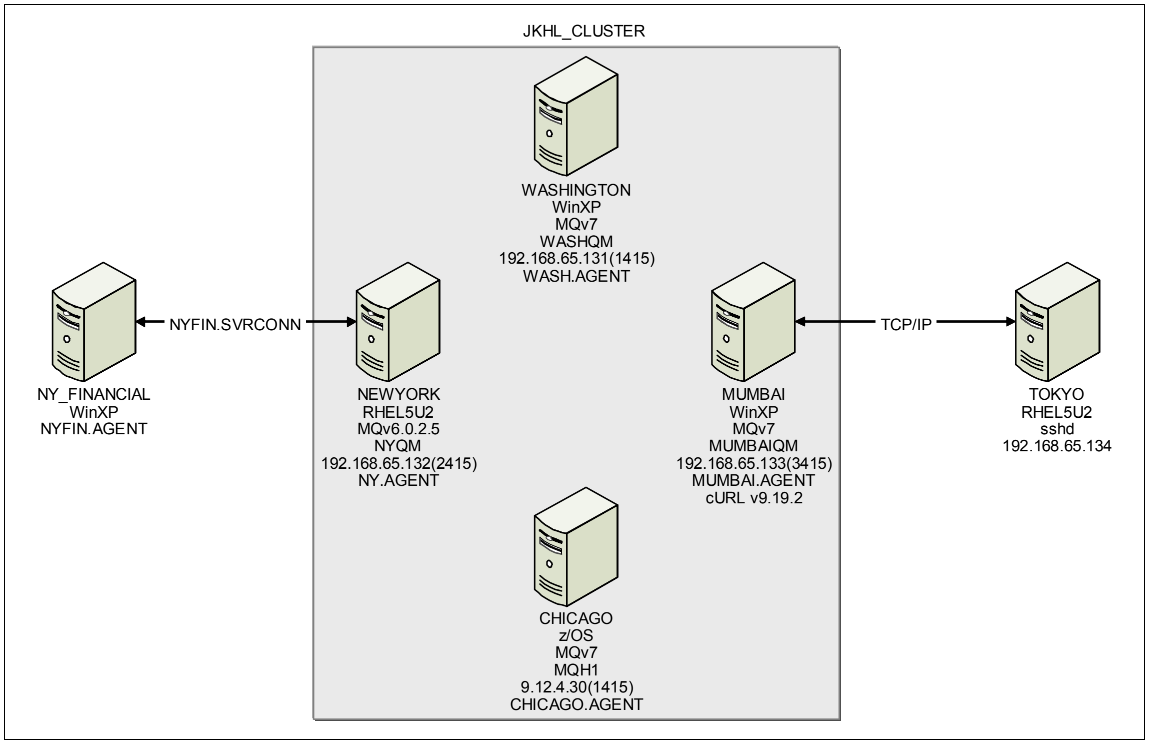


Figure 3-1 Redbooks topology

In this chapter, we set up the following Windows and Linux components that are required to support our WebSphere MQ File Transfer Edition file transfer scenarios. Effectively this means that we create all of the components shown in Figure 3-1 except for CHICAGO, which is implemented in WebSphere MQ File Transfer Edition on z/OS.

The Windows and Linux components that we configure in this chapter are:

- WebSphere MQ cluster configuration
- WebSphere MQ client channel configuration
- WebSphere MQ File Transfer Edition Server V7.0.1 installation

- WebSphere MQ File Transfer Edition Client V7.0.1 installation
- Apache Ant v1.7.0
- cURL V7.19.2 installation

Figure 3-2 shows the starting point for our installation and configuration process.

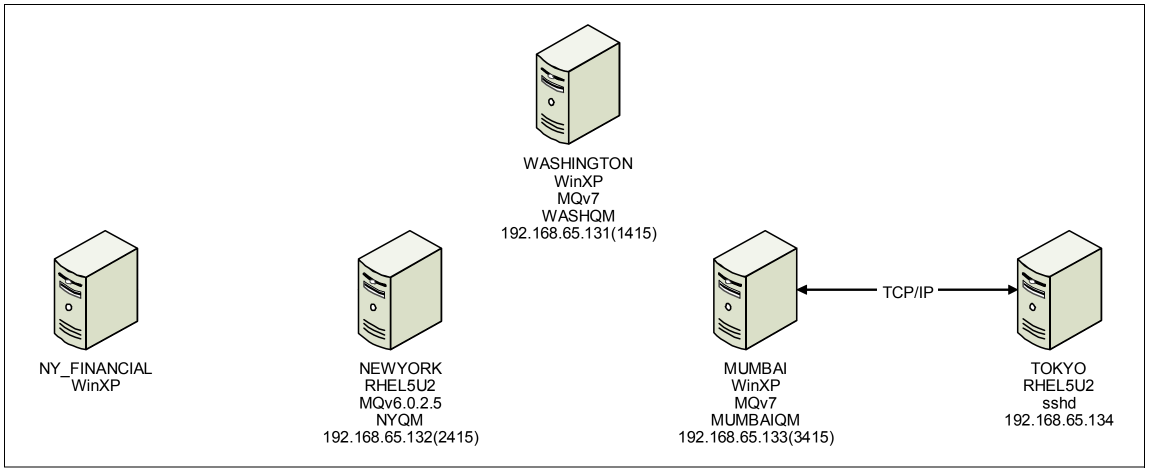


Figure 3-2 JKHL machines before installation and configuration for Redbooks scenario

Our installation and configuration process is to:

1. Create an MQ cluster that includes the three machines with MQ servers: WASHINGTON, NEWYORK, and, MUMBAI. These machines can then communicate with each other using the MQ messaging infrastructure.
2. Install FTE Server agents on WASHINGTON, NEWYORK, and, MUMBAI. WASHINGTON's queue managers:
   - WASHQM is configured as the FTE coordination queue manager.
   - NYQM and MUMBAIQM are configured as remote queue managers on the WASHQM machine.

   File transfers are now possible between these three machines.
3. Install FTE Tools on the WASHINGTON coordination queue manager machine.
4. Configure a client connection channel on the NEWYORK machine's NYQM queue manager, which allows NY_FINANCIAL to connect to NYQM using an FTE client that we install on NY_FINANCIAL

5. Install an FTE agent on NY_FINANCIAL. This agent uses the client connection channel that we created in step 4 to connect to NEWYORK's queue manager. This connection allows NY_FINANCIAL to participate in file transfers with the other machines that have FTE agents.
6. Install cURL on the MUMBAI machine. Currently, MUMBAI uses FTP to provide JKHLs with a manual FTP connection to their newly acquired Tokyo office. Installing cURL on the MUMBAI machine allows FTE to use Apache Ant to automate and centrally manage file transfers to and from TOKYO.

3.2 WebSphere MQ configuration

In this section, the aspects of the WebSphere MQ configuration that we discuss are:

- WebSphere MQ cluster configuration
- WebSphere MQ channel configuration
- WebSphere MQ client channel configuration
- Security

3.2.1 WebSphere MQ cluster configuration

The foundation of our JKHL Travel scenario is a WebSphere MQ (MQ) messaging infrastructure. To create this infrastructure we installed MQ on two Windows XP machines and one RedHat Linux machine (RHEL5U2). We then configured MQ communications channels between these machines.

The installation tasks that we completed are:

- On the Windows XP machines, we installed WebSphere MQ V7

  For detailed information about how to accomplish an MQ install on version 7, see the *Quick Beginnings* sections at:

  `http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

- On the Linux machines, we installed WebSphere MQ V6.0.2.5.

  For detailed information about how to accomplish an MQ install on version 6, see the *Quick Beginnings* sections at:

  `http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp`

As shown in Figure 3-3 on page 37, we chose to configure our three JKHL MQ machines as a cluster. This architecture has the benefits of simplicity, flexibility, ease of administration, and load balancing in a production environment.

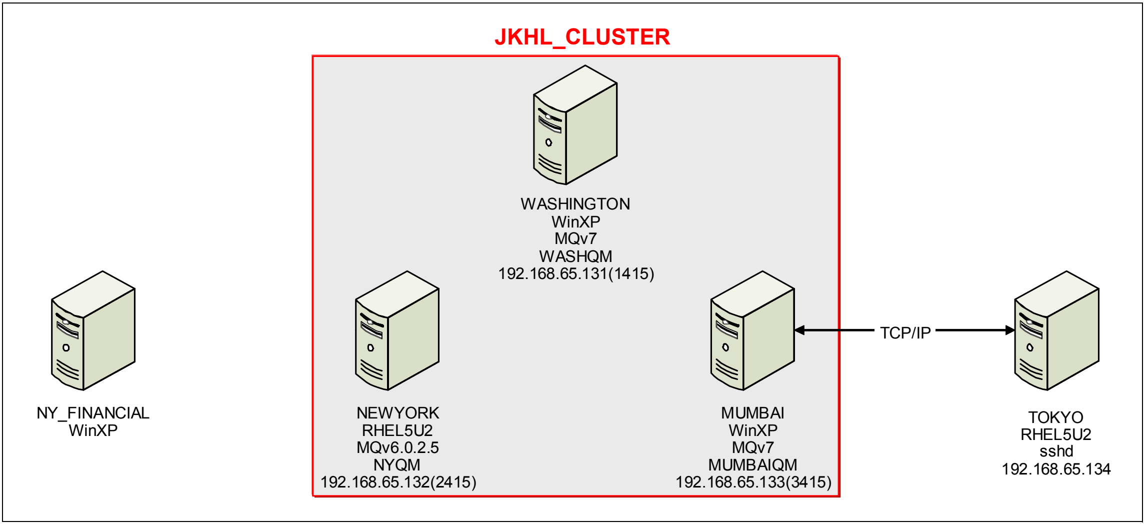


Figure 3-3 MQ cluster configured

More detailed information about how to set up an MQ v7 cluster is at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzah.doc/qc10120_.htm

For setting up a cluster for WebSphere MQ V6, see:

http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzah.doc/qc10120_.htm

A cluster configuration can be accomplished using either the MQ Explorer GUI or using MQSC command line scripts. Example 3-1 is an example of such a script.

Example 3-1 MQSC script for creating NEWYORK's cluster definitions

```
*---------------------------------------------------
* Setup Qmgr
*---------------------------------------------------
alter qmgr +
      repos(JKHL_CLUSTER)
*---------------------------------------------------
* Cluster Receiver Channel
*---------------------------------------------------
define channel(TO.NY) +
       chltype(clusrcvr)  +
       cluster(JKHL_CLUSTER) +
       conname('newyork(2415)') +
       replace
```

```
*--------------------------------------------------
* Cluster Sender Channel to WASHINGTON
*--------------------------------------------------
define channel(TO.WASH) +
       chltype(clussdr)  +
       cluster(JKHL_CLUSTER) +
       conname('washington(1415)') +
       replace
```

3.2.2 WebSphere MQ channel configuration

Although we did not use this configuration for our scenario, MQ can also be configured for FTE by using MQ channels, as shown in Figure 3-4.

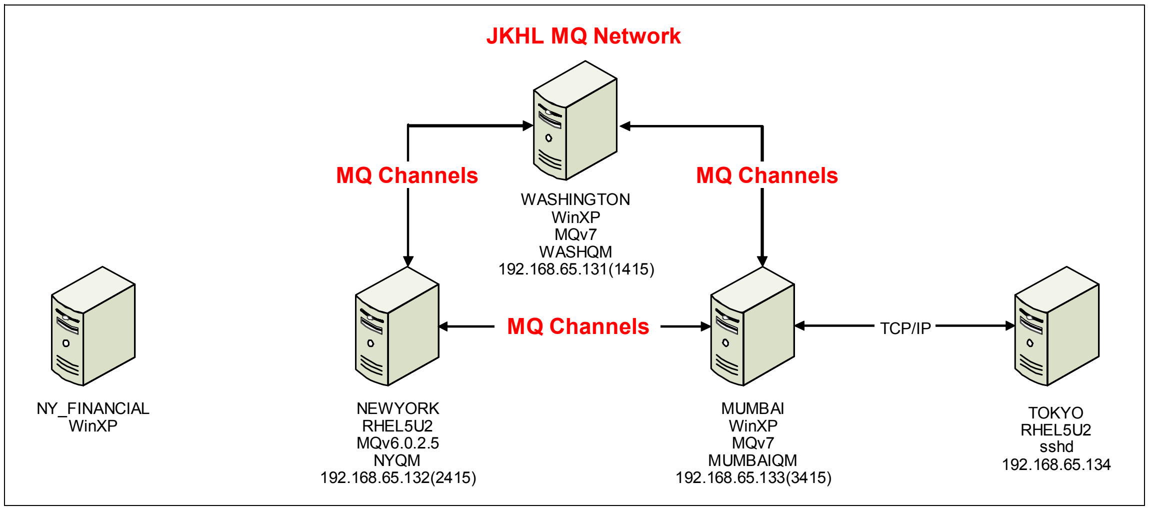


Figure 3-4 Alternative MQ architecture that uses sender and receiver channels

Example 3-2 shows the configuration script for NEWYORK's MQ channels for communication with WASHINGTON.

Example 3-2 mqsc script to create sender and receiver channels to NEWYORK

```
DEFINE QLOCAL(NYQM) +
   USAGE(xmitq)     +
   REPLACE
DEFINE CHANNEL(WASHQM.NYQM) +
   CHLTYPE(sdr)     +
   XMITQ(NYQM)      +
   TRPTYPE(tcp)     +
   CONNAME('newyork(2415)') +
   DISCINT(999999)   +
   REPLACE
DEFINE CHANNEL(NYQM.WASHQM) +
   CHLTYPE(rcvr)     +
   TRPTYPE(tcp)      +
   REPLACE
ALTER QMGR CHAD(ENABLED)
```

3.2.3 WebSphere MQ client channel configuration

JKHL Travel uses the NY_FINANCIAL machine in JKHL Travel's New York office to process financial analysis and pricing. To fulfill this role, NY_FINANCIAL must be connected to JKHL's FTE infrastructure. Because NY_FINANCIAL does not have an MQ server, we must allow it to connect to NEWYORK as an FTE client, which we can do by creating the NYFIN.SVRCONN client connection channel, as shown in Figure 3-5.

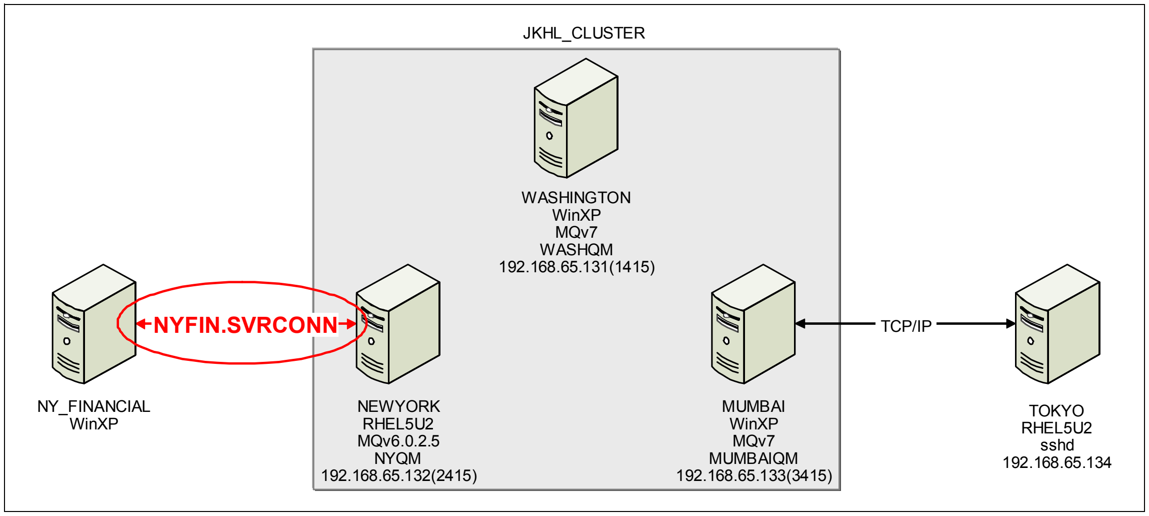


Figure 3-5 Client channel configured

From the command line on the NEWYORK machine, we used the mqsc script in Example 3-3.

Example 3-3 mqsc script for creating the NYFIN.SVRCONN channel

```
DEFINE CHANNEL(NYFIN.SVRCONN) CHLTYPE(SVRCONN) +
        TRPTYPE(TCP) REPLACE +
   DESCR('Channel for NY_FINANCIAL client') +
        MCAUSER('wmbadmin')  +
        MAXMSGL(4194304)
```

Using this channel, we can complete the FTE client installation that we described in 3.5, “Installing the FTE client agent” on page 62.

3.2.4 Security

Due to JKHL’s requirement to transmit sensitive information over their MQ infrastructure, we enabled Secure Socket Layer (SSL) security.

Detailed information about how to set up SSL is covered in the following documentation.

- For WebSphere MQ V7

  http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzas.doc/sy10120_.htm

- For WebSphere MQ V6

  http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzas.doc/sy10120_.htm

We discuss SSL configuration for the FTE client at NY_FINANCIAL in detail in section 7.4, “Enabling SSL security on FTE client agents” on page 171.

3.3 Installing the FTE server

In this section, we describe how to install the WebSphere MQ File Transfer Edition server coordination queue manager and the WebSphere MQ File Transfer Edition remote queue managers.

3.3.1 Installing the server on the coordination queue manager

In this section, we cover how to install the FTE server on the machine that we used in our scenario as the coordination queue manager, WASHINGTON. Our goal in this installation is to create the agent WASH.AGENT on the queue manager, WASHQM, as shown in Figure 3-6 on page 41. We created this FTE server on Windows XP.

Coordination queue manager: For security reasons in a production environment, no agents are installed on the coordination queue manager. Typically, only administrators have access to the coordination queue manager.

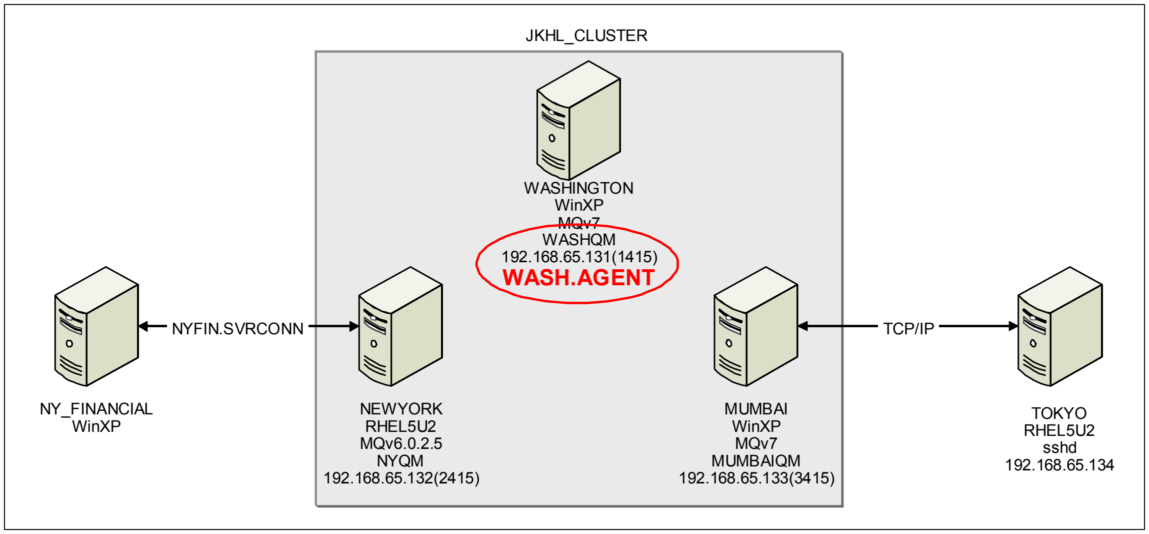
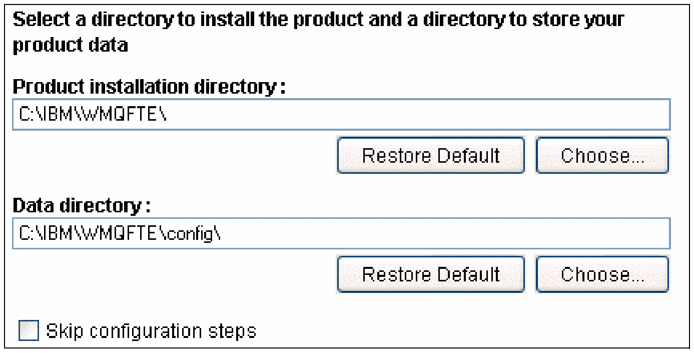


Figure 3-6 FTE agent installed on coordination queue manager

The steps for accomplishing this installation are:

1. Start WASHQM.
2. Locate your WebSphere MQ File Transfer Edition server install files.
3. Initiate the FTE server install by running install.exe in the Windows installation directory.
4. Continue with the install wizard, and then accept the licensing agreement.
5. Specify a directory in which to install FTE, as shown in Figure 3-7 on page 42.


Select a directory to install the product and a directory to store your product data

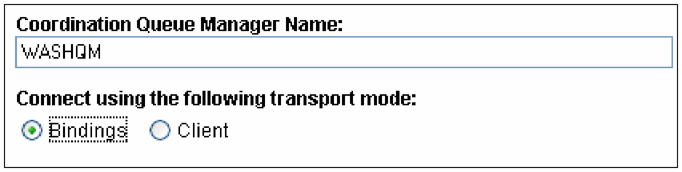
Product installation directory: C:\IBM\WMQFTE\

Data directory: C:\IBM\WMQFTE\config\

☐ Skip configuration steps

Figure 3-7 Specify FTE server install directories

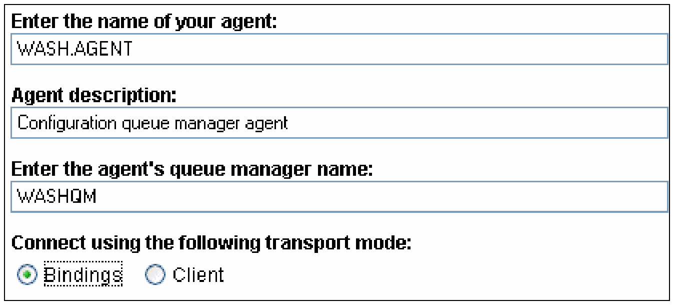
6. Enter the name of the coordination queue manager, and specify the bindings mode, as shown in Figure 3-8. The queue manager that you name here should already exist. Because the coordination queue manager is local, you can specify bindings mode.


Coordination Queue Manager Name: WASHQM

Connect using the following transport mode: [✓ Bindings] Client

Figure 3-8 Specify the name of an existing queue manager

7. Enter agent queue manager details. You can specify any desired name for your agent, as shown in Figure 3-9. The agent's queue manager name must refer to an existing queue manager. Because the agent's queue manager is local, you can specify the bindings mode.


Enter the name of your agent: WASH.AGENT

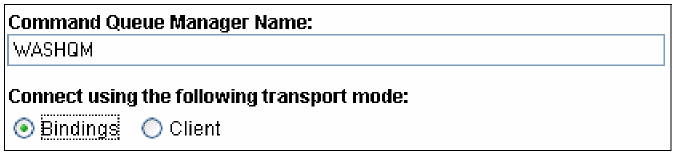
Agent description: Configuration queue manager agent

Enter the agent's queue manager name: WASHQM

Connect using the following transport mode: [✓ Bindings] Client

Figure 3-9 Enter agent queue manager details

8. Specify the name of an existing queue manager that you want to use as your command queue manager, as shown in Figure 3-10. Because the command queue manager is local, you can specify bindings mode.

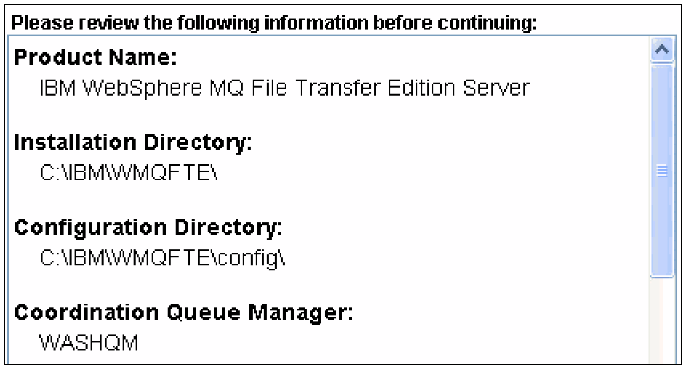


Command Queue Manager Name:
WASHQM

Connect using the following transport mode:
[✓ Bindings] Client

Figure 3-10 Specify the name of the command queue manager

9. Review the install options, as shown in Figure 3-11. You must scroll down to see all of the options that you specified. Click **Install**.



Please review the following information before continuing:

Product Name:
IBM WebSphere MQ File Transfer Edition Server

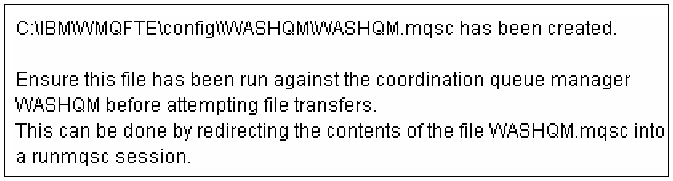
Installation Directory:
C:\IBM\WMQFTE\

Configuration Directory:
C:\IBM\WMQFTE\config\

Coordination Queue Manager:
WASHQM

Figure 3-11 Review installation options

10. The first phase of the install completes with the appearance of the panel shown in Figure 3-12. Do not continue until you run the MQSC commands, as shown in Example 3-4 on page 44.



C:\IBM\WMQFTE\config\WASHQM\WASHQM.mqsc has been created.

Ensure this file has been run against the coordination queue manager WASHQM before attempting file transfers.
This can be done by redirecting the contents of the file WASHQM.mqsc into a runmqsc session.

Figure 3-12 This script must now be run from the command line

WASHQM: The queue manager, WASHQM, must be running.

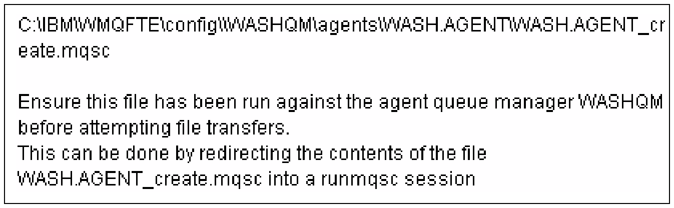
11. Use the MQ utility, runmqsc, to run the WASHQM.mqsc script from the command line, as shown in Example 3-4. Make sure that there are no errors at the bottom of the script output.

Example 3-4 Running the coordination queue manager script

```
C:\IBM\WMQFTE\config\WASHQM>runmqsc WASHQM < WASHQM.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2008.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager WASHQM.

     1 : DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
AMQ8690: WebSphere MQ topic created.
     2 : ALTER TOPIC('SYSTEM.FTE') NPMSGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)
AMQ8691: WebSphere MQ topic changed.
     3 : DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM)
REPLACE
AMQ8006: WebSphere MQ queue created.
     4 : ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for WMQFTE Pub/Sub
interface')
AMQ8008: WebSphere MQ queue changed.
       : * Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST
       : * Value prior to alteration:
     5 : DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
AMQ8550: Display namelist details.
   NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
   NAMCOUNT(3)
   NAMES(SYSTEM.BROKER.DEFAULT.STREAM
        ,SYSTEM.BROKER.ADMIN.STREAM
        ,SYSTEM.FTE)
   DESCR(A list of queues for the queued Pub/Sub interface to monitor)
   ALTDATE(2009-04-08)                    ALTTIME(21.05.12)
     6 : ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +
       :  NAMES(SYSTEM.BROKER.DEFAULT.STREAM+
       :  ,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
AMQ8551: WebSphere MQ namelist changed.
       : * Altering PSMODE.  Value prior to alteration:
     7 : DISPLAY QMGR PSMODE
AMQ8408: Display Queue Manager details.
   QMNAME(WASHQM)                         PSMODE(ENABLED)
     8 : ALTER QMGR PSMODE(ENABLED)
AMQ8005: WebSphere MQ queue manager changed.
8 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

12. After you successfully run the coordination queue manager script, return to the install wizard, and click **Next**.

13. You must now run a script to create an FTE agent. Figure 3-13 shows the request to run the script that creates an FTE agent.



C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT\WASH.AGENT_cr
eate.mqsc

Ensure this file has been run against the agent queue manager WASHQM before attempting file transfers.
This can be done by redirecting the contents of the file WASH.AGENT_create.mqsc into a runmqsc session

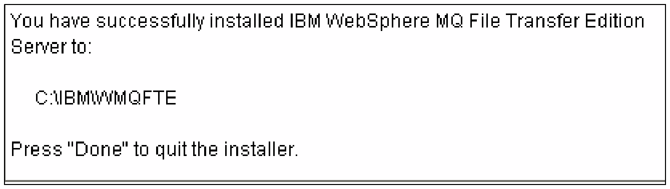
Figure 3-13 Request to run the mqsc script that will create an FTE agent

14. Use the MQ utility, runmqsc, to run the WASH.AGENT_create.mqsc script, which we show in Example 3-5. Make sure that there are no errors at the bottom of the script output.

Example 3-5 Running the create agent script

```
C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT>runmqsc WASHQM <
WASH.AGENT_create.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2008.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager WASHQM.

     1 : DEFINE QLOCAL(SYSTEM.FTE.COMMAND.WASH.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     2 : DEFINE QLOCAL(SYSTEM.FTE.DATA.WASH.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     3 : DEFINE QLOCAL(SYSTEM.FTE.REPLY.WASH.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     4 : DEFINE QLOCAL(SYSTEM.FTE.STATE.WASH.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     5 : DEFINE QLOCAL(SYSTEM.FTE.EVENT.WASH.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
        :  SHARE +
        :  NOTRIGGER +
        :  USAGE(NORMAL) +
        :  REPLACE
AMQ8006: WebSphere MQ queue created.
5 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

15. After you run the create agent script and make sure that there were no errors, return to the install wizard, and click **Next**.
16. A message that the install is complete is displayed, as shown in Figure 3-14. To close the wizard, press **Done**.


You have successfully installed IBM WebSphere MQ File Transfer Edition Server to:

C:\IBM\WMQFTE

Press "Done" to quit the installer.

Figure 3-14 Installation complete

17. Open a command line window, change to the c:\IBM\WMQFTE\bin directory, and then start the agent, WASH.AGENT, as shown in Example 3-6.

Example 3-6 Start the agent, WASH.AGENT

```
C:\IBM\WMQFTE\bin>fteStartAgent WASH.AGENT
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0030I: The request to start agent 'WASH.AGENT' on this machine has
been submitted.
BFGCL0031I: Agent log files located at:
C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT
C:\IBM\WMQFTE\bin>
```

FTE bin directory: We suggest that you add the FTE bin directory, “C:\IBM\WMQFTE\bin”, to the path system variable, which simplifies accessing FTE commands from the command line.

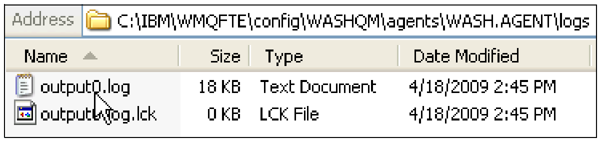
18. Ping the agent to confirm that it started, as shown in Example 3-7.

Example 3-7 Ping the agent

```
C:\IBM\WMQFTE\bin>ftePingAgent WASH.AGENT
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
```

```
BFGCL0212I: Issuing ping request to agent WASH.AGENT
BFGCL0193I: The request has been received by the source agent.
BFGCL0213I: agent WASH.AGENT responded to ping in 0.469 seconds.
```

19. Open the agent log, as shown in Figure 3-15.


Address C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT\logs

Name	Size	Type	Date Modified
output0.log	18 KB	Text Document	4/18/2009 2:45 PM
output0.log.lck	0 KB	LCK File	4/18/2009 2:45 PM

Figure 3-15 WASH.AGENT log: output0.log

20. Examine the contents of the log, as shown in Example 3-8.

Example 3-8 Contents of WASH.log

```
************ Start Display Current Environment ************
Build level: f000-20090407-1353
Java runtime version:
    J2RE 1.5.0 IBM J9 2.3 Windows XP x86-32 j9vmwi3223-20080710 (JIT
enabled)
    J9VM - 20080625_20583_lHdSMr
    JIT  - 20080620_1845_r8
    GC   - 200806_19
Properties:
    agentDesc=Washington's Agent, agentName=WASH.AGENT,
agentQMgr=WASHQM, commandPath=c:/curl
    coordinationQMgr=WASHQM
************* End Display Current Environment *************
[18/04/2009 14:54:32:343 EDT] 00000001 AgentRuntime  I   BFGAG0058I:
The agent has been successfully initialized.
[18/04/2009 14:54:33:859 EDT] 00000001 AgentRuntime  I   BFGAG0059I:
The agent has been successfully started.
```

3.3.2 Installing servers on remote queue managers

In this section, we discuss how to install the FTE server on the machines that we used in our scenario as remote FTE servers: NEWYORK and MUMBAI.

Agent and command queue managers: In our scenario, for simplicity we use the same queue manager for both the agent and the command queue manager. In a production environment, for security reasons, the command queue manager does not have an agent. Typically, users (including applications) have access to the command queue manager. Whereas the agent queue manager is configured to secure specific resources.

Installing the FTE server on MUMBAI using the GUI installer

In this section, we cover installing the FTE server on a Windows XP machine, which we used in our scenario as a remote FTE server: MUMBAI. Our goal in this installation is to create the agent MUMBAI.AGENT on the queue manager MUMBAIQM. Figure 3-16 illustrates the installation.

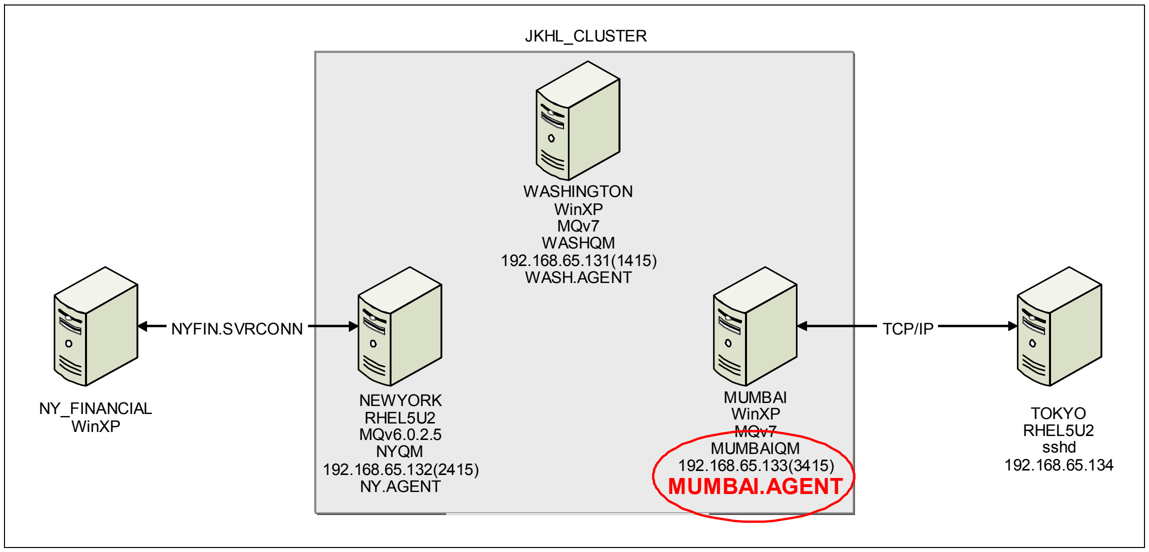
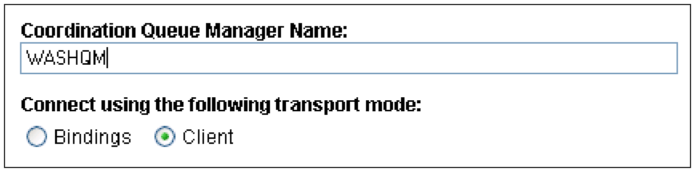


Figure 3-16 Installing the MUMBAI.AGENT machine

The following lists contains primarily those steps that are different from the previously described FTE server installation on WASHINGTON, the coordination queue manager machine.

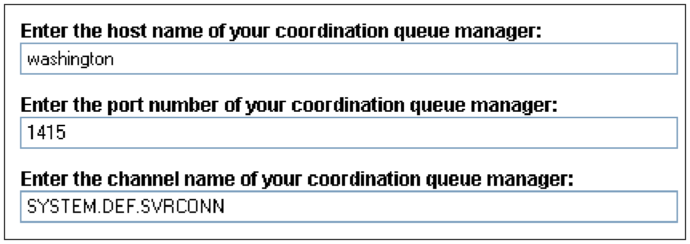
1. Start MUMBAIQM.
2. Locate your WebSphere MQ File Transfer Edition server install files.
3. Initiate the FTE server install by going to the Windows installation directory, and running install.exe.
4. Continue with the install wizard. Specify the same installation directory that you specified for the WASHINGTON FTE server: C:\IBM\WMQFTE
5. At the Queue Manager panel, make the configurations as shown in Figure 3-17 on page 51. Because the coordination queue manager is not on MUMBAI, you must use client mode.


Coordination Queue Manager Name:
WASHQM

Connect using the following transport mode:
( ) Bindings [✓ Client]

Figure 3-17 Queue manager panel

6. Enter the coordination queue manager details, as shown in Figure 3-18.

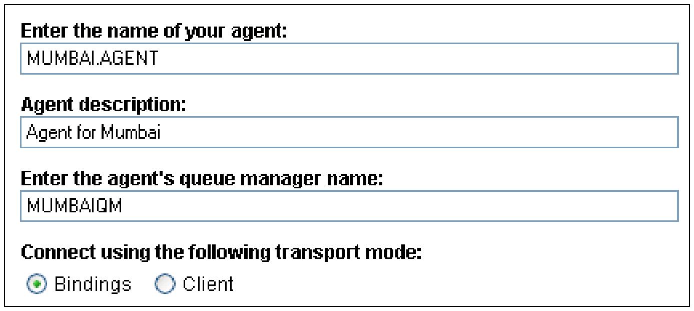

Enter the host name of your coordination queue manager:
washington

Enter the port number of your coordination queue manager:
1415

Enter the channel name of your coordination queue manager:
SYSTEM.DEF.SVRCONN

Figure 3-18 Coordination queue manager

7. Enter the agent queue manager details, as shown in Figure 3-19. Because the agent's queue manager is local, you can use bindings mode.


Enter the name of your agent:
MUMBAI.AGENT

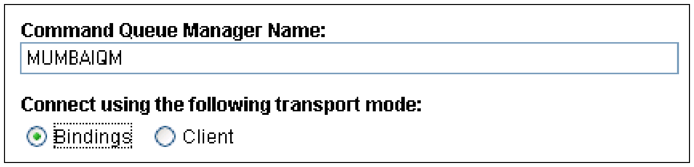
Agent description:
Agent for Mumbai

Enter the agent's queue manager name:
MUMBAIQM

Connect using the following transport mode:
[✓ Bindings] ( ) Client

Figure 3-19 Agent queue manager

8. Enter the command queue manager name. The command queue manager is local, so you can use the bindings mode, as shown in Figure 3-20.


Command Queue Manager Name:
MUMBAIQM

Connect using the following transport mode:
[✓ Bindings] ( ) Client

Figure 3-20 Command queue manager

9. Proceed with the installation until you arrive at the “Run on Coordination Queue Manager” panel. Do not run the script. Click **Next**.
10. At the “Run on Agent Queue Manager” panel, run the script to create the MUMBAI.AGENT.

FTE bin directory: We suggest that you add the FTE bin directory, “/home/wmbadmin/IBM/WMQFTE/bin”, to the PATH environment variable, which simplifies accessing FTE commands from the command line.

11. Finish the installation.
12. Start the MUMBAI.AGENT.
13. Ping the agent using the **`ftePingAgent`** command line command.
14. Examine the agent log.

Installing the FTE server on NEWYORK using the command line installer

In this section, we cover how to install the FTE server on a RedHat Linux machine, which we used in our scenario as a remote FTE server: NEWYORK. Our goal in this installation is to create the agent NY.AGENT on the queue manager NYQM. Figure 3-21 illustrates the installation.

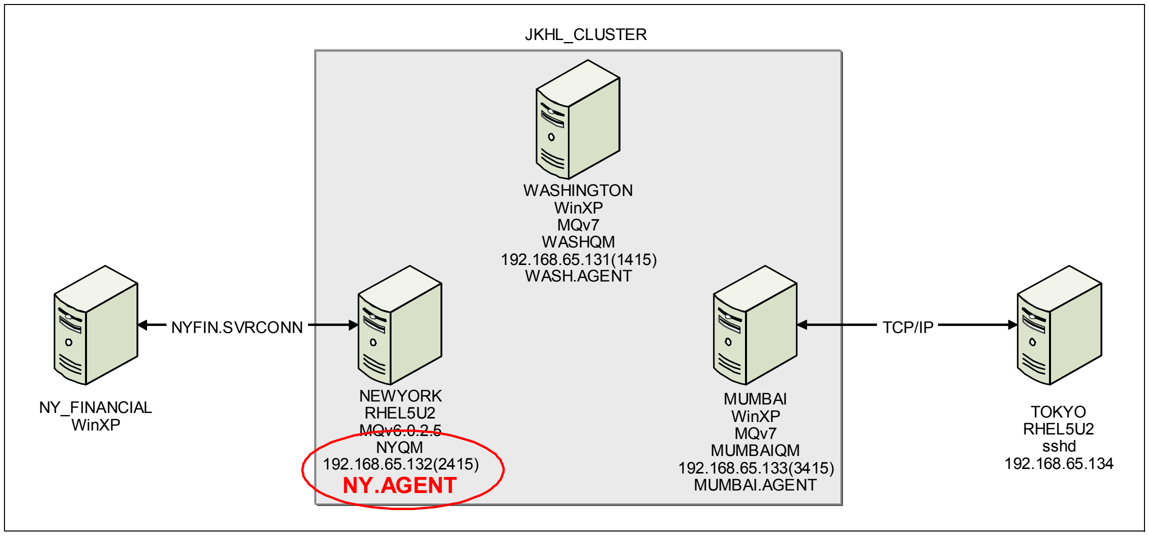


Figure 3-21 Installing the NY.AGENT machine

The following list contains primarily those steps that are different from the previously described FTE server installation on WASHINGTON, the coordination queue manager machine. We used the command line installer to show how you can install FTE on systems that do not allow you to use the FTE GUI installer.

1. Start NYQM.
2. Locate your WebSphere MQ File Transfer Edition server install files.
3. Example 3-9 shows the command to run a console command and create a response file.

Example 3-9 Install.bin

```
[wmbadmin@newyork CDROM_Installers]$
Disk1/InstData/Linux/VM/install.bin -i console -r
/home/wmbadmin/IBM/installer.properties
```

4. Choose your local language, as shown in Example 3-10, and press **Enter**.

Example 3-10 Selecting the language to use

```
Preparing to install...
Extracting the JRE from the installer archive...
Unpacking the JRE...
Extracting the installation resources from the installer archive...
Configuring the installer for this system's environment...

Launching installer...

Preparing CONSOLE Mode Installation...

=======================================================================
Choose Locale...
----------------

    1- Deutsch
  ->2- English
    3- Español
    4- Français
    5- Português  (Brasil)

CHOOSE LOCALE BY NUMBER:
```

5. Accept the license agreement.

6. Enter the installation directory, which we show in Example 3-11.

Example 3-11 Specify the installation directory

```
Product installation directory :
---------------------------------

Product installation directory :  (DEFAULT: /opt/IBM/WMQFTE)
   : /home/wmbadmin/IBM/WMQFTE

```

7. Accept the default choice of “Enter Configuration Steps”.
8. Specify the configuration directory, as shown in Example 3-12.

Example 3-12 Configuration directory

```
=======================================================================
Product configuration directory :
----------------------------------

Data directory :  (DEFAULT: /var/IBM/WMQFTE/config):
/home/wmbadmin/IBM/WMQFTE/config
```

9. Specify the coordination queue manager, as shown in Example 3-13.

Example 3-13 Coordination queue manager

```
=======================================================================
Get Coordination Queue Manager:
-------------------------------

Coordination Queue Manager : (DEFAULT: ): WASHQM

```

10. Select **Client** as the transport mode, as shown in Example 3-14. You choose this option because the coordination queue manager is not on the local machine.

Example 3-14 Transport mode

```
=======================================================================
Connect using the following transport mode:
-------------------------------------------

    1- Bindings
  ->2- Client
```

```
ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE
DEFAULT:
   :
```

11. Enter the coordination queue manager host name, as shown in Example 3-15.

Example 3-15 Coordination queue manager host name

```
=======================================================================
Coordination QMgr Host Name
---------------------------

Enter requested information

Enter the host name of your coordination queue manager (DEFAULT: )
   : washington
```

12. Enter the coordination queue manager port, as shown in Example 3-16.

Example 3-16 Coordination queue manager port

```
=======================================================================
Coordination QMgr Port
----------------------

Enter requested information

Enter the port number of your coordination queue manager. (DEFAULT:
1414)
   : 1415
```

13. Accept the default coordination queue manager channel, as shown in Example 3-17.

Example 3-17 Coordination queue manager channel

```
=======================================================================
Coordination QMgr Channel
-------------------------

Enter requested information

Enter the channel name of your coordination queue manager. (DEFAULT:
   SYSTEM.DEF.SVRCONN):
```

14. Enter the agent name, description, and queue manager, as shown in Example 3-18.

Example 3-18 Agent details

```
===============================================================================
Agent Name
----------

Enter requested information

Enter the name of your agent:  (DEFAULT: ): NY.AGENT

===============================================================================
Agent Description
-----------------

Enter requested information

Agent description (DEFAULT: ): New York's agent

===============================================================================
Agent QMgr
-----------

Enter requested information

Enter the agent's queue manager name: (DEFAULT: WASHQM): NYQM
```

15. Enter the transport mode for the agent’s connection to its command queue manager, as shown in Example 3-19. Because its command queue manager is local, select **bindings** mode.

Example 3-19 Transport mode

```
===============================================================================
Transport mode
--------------

Connect using the following transport mode:

    1- Bindings
  ->2- Client
```

```
ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE
DEFAULT:
   : 1
```

16. Specify the command queue manager name (accept the default), as shown in Example 3-20. Select **Bindings** as the transport mode.

Example 3-20 Connection method

```
===============================================================================
Command QMgr Name
-----------------

Command Queue Manager Name: (DEFAULT: NYQM):

===============================================================================
Command QMgr connection method
------------------------------

Connect using the following transport mode:

    1- Bindings
  ->2- Client

ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE
DEFAULT:
   : 1
```

17. Review the installation options, and then continue with the install.

18. Do not run the “WASHQM.mqsc” script because that script ran when you installed the coordination queue manager. Press **Enter**. See Example 3-21.

Example 3-21 Run on coordination queue manager

```
===============================================================================
Run on Coordination Queue Manager
---------------------------------

/home/wmbadmin/IBM/WMQFTE/WASHQM/WASHQM.mqsc has been created.

Ensure this file has been run against the coordination queue manager
WASHQM
before attempting file transfers.
```

```
This can be done by redirecting the contents of the file WASHQM.mqsc
into a
runmqsc session.

PRESS <ENTER> TO CONTINUE:
```

19. Stop at the "Run on Agent Queue Manager" panel, which we show in Example 3-22.

Example 3-22 Run on agent queue manager

```
===============================================================================
Run on Agent Queue Manager
--------------------------

/home/wmbadmin/IBM/WMQFTE/WASHQM/agents/NY.AGENT/NY.AGENT_create.mqsc

Ensure this file has been run against the agent queue manager NYQM
before
attempting file transfers.
This can be done by redirecting the contents of the file
NY.AGENT_create.mqsc
into a runmqsc session

PRESS <ENTER> TO CONTINUE:
```

20. Run the script against NYQM, as shown in Example 3-23.

Example 3-23 Running script for NYQM

```
[wmbadmin@newyork NY.AGENT]$ runmqsc NYQM < NY.AGENT_create.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager NYQM.

     1 : DEFINE QLOCAL(SYSTEM.FTE.COMMAND.NY.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     2 : DEFINE QLOCAL(SYSTEM.FTE.DATA.NY.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     3 : DEFINE QLOCAL(SYSTEM.FTE.REPLY.NY.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     4 : DEFINE QLOCAL(SYSTEM.FTE.STATE.NY.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
     5 : DEFINE QLOCAL(SYSTEM.FTE.EVENT.NY.AGENT) +
       :  DEFPRTY(0) +
       :  DEFSOPT(SHARED) +
       :  GET(ENABLED) +
       :  MAXDEPTH(5000) +
       :  MAXMSGL(4194304) +
       :  MSGDLVSQ(PRIORITY) +
       :  PUT(ENABLED) +
       :  RETINTVL(999999999) +
       :  SHARE +
       :  NOTRIGGER +
       :  USAGE(NORMAL) +
       :  REPLACE
AMQ8006: WebSphere MQ queue created.
5 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

21. After the script runs without errors, press **Continue**. Exit the installer.
22. Start the agent, ping it, and then examine its log file.

Unattended installations

In addition to the GUI and command modes of installation, WebSphere MQ FIle Transfer Edition provides an unattended or silent mode. This mode uses an installer.properties file that you can create from either a previous console or manually.

More details about this mode of installation are in the WebSphere MQ File Transfer Edition Information Center.

FTE server agents installed on all MQ servers

Figure 3-22 on page 61 shows the finished result of installing the FTE server on our three server machines. Each of these machines is located in a JKHL Travel office.

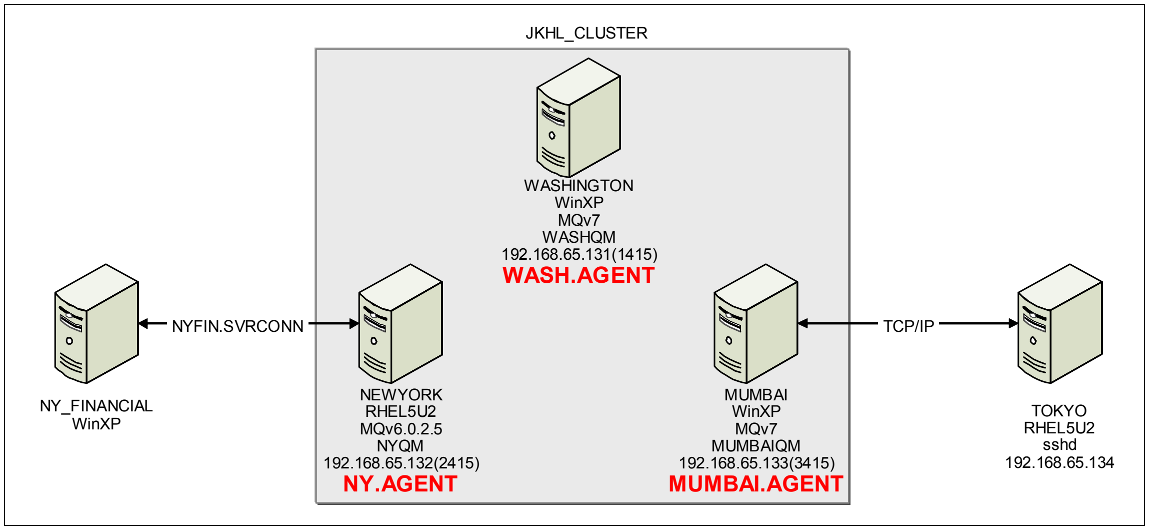


Figure 3-22 FTE agents installed on all MQ server machines

3.4 Installing the FTE Tools

In this section, we cover how to install FTE Tools on the machine that we used in our scenario as the coordination queue manager, WASHINGTON. Our goal in this installation is to add the Eclipse plug-in, Figure 3-23, to the WebSphere MQ Explorer, which allows us to manage and monitor file transfers throughout the JKHL MQ infrastructure.

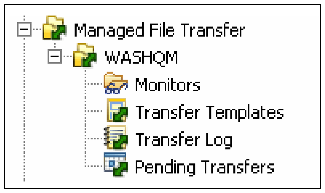
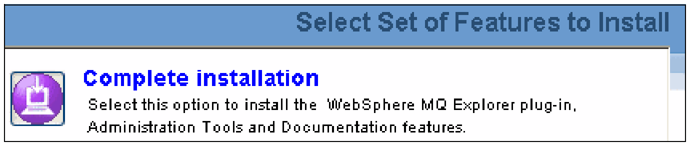


Figure 3-23 FTE Tools installed in WebSphere MQ Explorer

The steps for this installation are:

1. Stop the WebSphere MQ Explorer.
2. Locate your WebSphere MQ File Transfer Edition tools install files.
3. Initiate the FTE server install by running install.exe in Windows installation directory.
4. Continue with the install wizard, and then accept the licensing agreement.

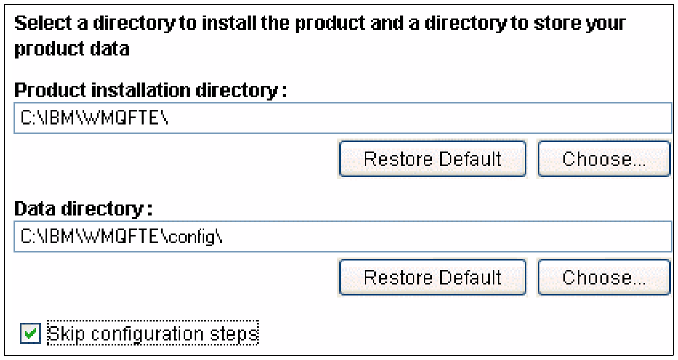
5. On the “Select Set of Features to Install” panel, click **Complete Installation**, as shown in Figure 3-24, and then click **Next**.



Select this option to install the WebSphere MQ Explorer plug-in, Administration Tools and Documentation features.

Figure 3-24 Complete installation

6. Specify a directory in which to install FTE, as shown in Figure 3-25.



Select a directory to install the product and a directory to store your product data

Product installation directory: C:\IBM\WMQFTE\

Data directory: C:\IBM\WMQFTE\config\

✓ Skip configuration steps

Figure 3-25 Specify a directory

7. Review the install options, and then press **Install**.
8. A message is displayed that lets you know that the tools install is complete. To close the wizard, click **Done**.
9. Start the WebSphere MQ Explorer. You should now see the Managed File Transfer Tools that are shown in Figure 3-23 on page 61.

3.5 Installing the FTE client agent

In this section, we describe how to create the FTE client agent NYFIN.AGENT, which Figure 3-26 on page 63 illustrates.

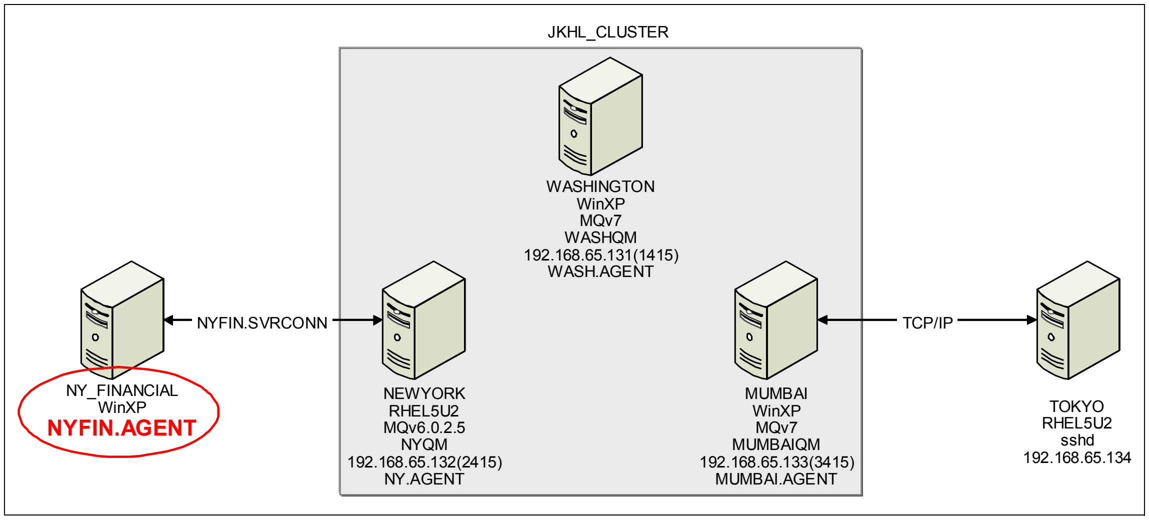
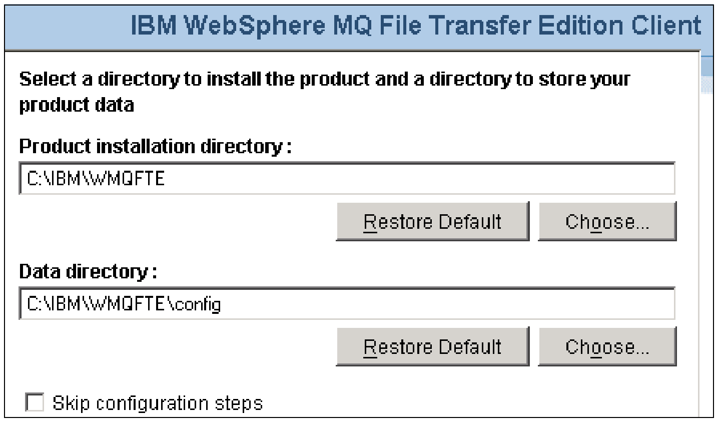


Figure 3-26 FTE client agent installed

To install the FTE agent:

1. Locate your WebSphere MQ File Transfer Edition client install files.
2. Initiate the FTE server install by running install.exe in the Windows installation directory.
3. Continue with the install wizard. Accept the licensing agreement. Specify a directory in which to install FTE, as shown in Figure 3-27.



IBM WebSphere MQ File Transfer Edition Client

Select a directory to install the product and a directory to store your product data

Product installation directory :

C:\IBM\WMQFTE

Data directory :

C:\IBM\WMQFTE\config

☐ Skip configuration steps

Figure 3-27 Specify installation directory

4. Enter the name of an existing coordination queue manager, as shown in Figure 3-28.

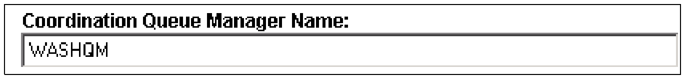
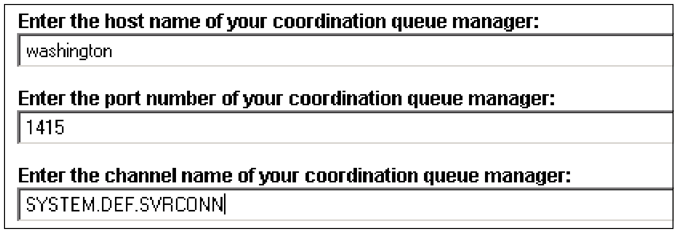

Coordination Queue Manager Name:
WASHQM

Figure 3-28   Specify the name of an existing coordination queue manager

5. Enter coordination queue manager details, as shown in Figure 3-29. This entry references the coordination queue manager that you previously defined.

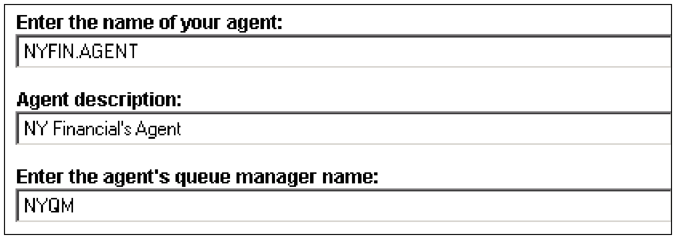

Enter the host name of your coordination queue manager:
washington

Enter the port number of your coordination queue manager:
1415

Enter the channel name of your coordination queue manager:
SYSTEM.DEF.SVRCONN

Figure 3-29   Coordination queue manager details

6. Enter the name of your client agent and the queue manager that it will use, as shown in Figure 3-30.

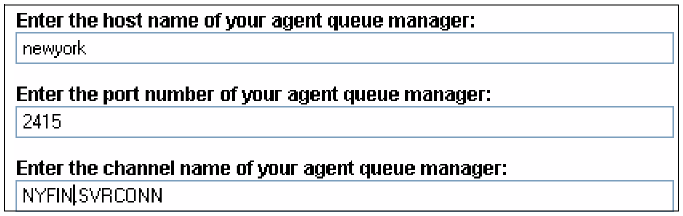

Enter the name of your agent:
NYFIN.AGENT

Agent description:
NY Financial's Agent

Enter the agent's queue manager name:
NYQM

Figure 3-30   Client agent details

7. Enter agent queue manager details, as shown in Figure 3-31.

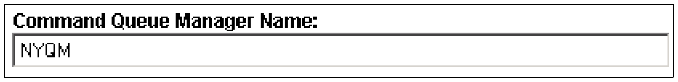

Enter the host name of your agent queue manager:
newyork

Enter the port number of your agent queue manager:
2415

Enter the channel name of your agent queue manager:
NYFIN.SVRCONN

Figure 3-31   Client agent queue manager details

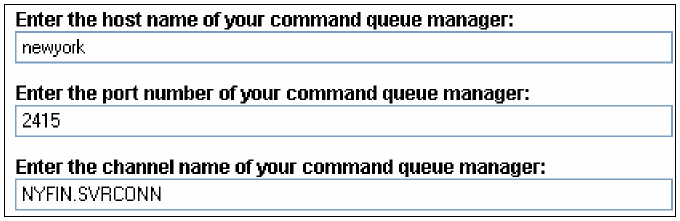
8. Specify the name of an existing queue manager that you want to use as your command queue manager, as shown in Figure 3-32.



Command Queue Manager Name:
NYQM

Figure 3-32   Client agent's command queue manager name

9. Enter command queue manager details, as shown in Figure 3-33.



Enter the host name of your command queue manager:
newyork

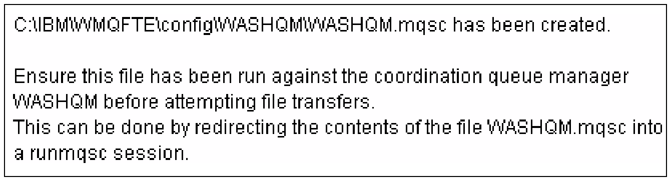
Enter the port number of your command queue manager:
2415

Enter the channel name of your command queue manager:
NYFIN.SVRCONN

Figure 3-33   Command queue manager details

10. Review the install options, and then click **Install**.

11. The first phase of the install completes with the appearance of the panel shown in Figure 3-34. Because you previously ran the coordination queue manager script (when you installed the FTE server on the coordination queue manager), you can skip this panel. Click **Next.**



C:\IBM\WMQFTE\config\WASHQM\WASHQM.mqsc has been created.

Ensure this file has been run against the coordination queue manager WASHQM before attempting file transfers.
This can be done by redirecting the contents of the file WASHQM.mqsc into a runmqsc session.

Figure 3-34   Skip the "Run on Coordination Queue Manager" step

12. Copy the NYFIN.AGENT_create script to the FTE client's queue manager machine (NEWYORK). Use the MQ utility, runmqsc, to run the NYFIN.AGENT_create.mqsc script from the command line, as shown in Figure 3-35 on page 66. Make sure that there are no errors at the bottom of the script output.

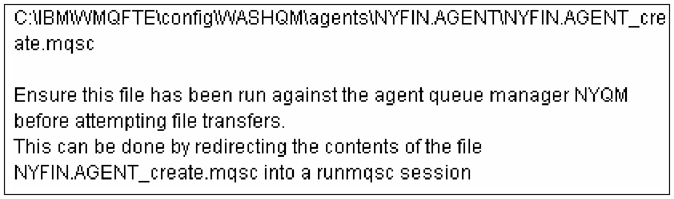

```
C:\IBM\WMQFTE\config\WASHQM\agents\NYFIN.AGENT\NYFIN.AGENT_cre
ate.mqsc

Ensure this file has been run against the agent queue manager NYQM
before attempting file transfers.
This can be done by redirecting the contents of the file
NYFIN.AGENT_create.mqsc into a runmqsc session
```

Figure 3-35 Running the MQSC script

13. After you run the create agent script and make sure that there are no errors, return to the install wizard, and click **Next**.
14. A message appears letting you know that the client install is complete. To close the wizard, click **Done**.

3.6 FTE properties files

WebSphere MQ File Transfer Edition uses several properties files that contain key information about your setup and are required for operation. The values that are defined in these properties files are used as the default parameters for all WebSphere MQ File Transfer Edition commands, unless you explicitly specify a different value on the command line.

As an example, in this section we show all of the properties that were set for the FTE server on NEWYORK:

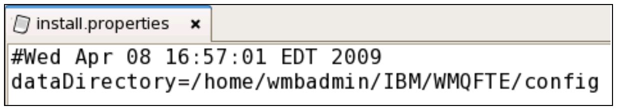
- install.properties
- wmqfte.properties
- command.properties
- coordination.properties, and
- agent.properties

After reviewing the default properties that were set, we must add an additional property to the agent.properties file. This additional property is necessary to allow fteAnt to use cURL to accomplish an SFTP connection to TOKYO. See Chapter 9, "Phase 3: Complex transfers" on page 227 for details.

3.6.1 install.properties

The install.properties file contains one entry: the path to the FTE data directory. This is a parameter that was set during the install.

This properties file is located in the FTE installation directory. On NEWYORK, it is located in the /wmbadmin/IBM/WMQFTE/ directory. Figure 3-36 shows its contents.



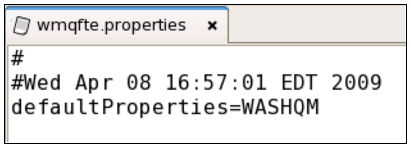
```
#Wed Apr 08 16:57:01 EDT 2009
dataDirectory=/home/wmbadmin/IBM/WMQFTE/config
```

Figure 3-36 install.properties

3.6.2 wmqfte.properties

The wmqfte.properties file contains one entry: the name of the default coordination queue manager.

This properties file is located in the /wmbadmin/IBM/WMQFTE/config directory. Figure 3-37 shows its contents.



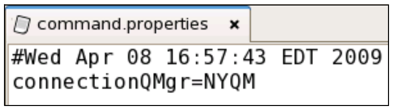
```
#
#Wed Apr 08 16:57:01 EDT 2009
defaultProperties=WASHQM
```

Figure 3-37 wmqfte.properties

3.6.3 command.properties

The command.properties file specifies the primary queue manager to connect to when issuing FTE commands (some commands also connect to the coodination queue manager).

This properties file is located in the /wmbadmin/IBM/WMQFTE/config/WASHQM directory. Figure 3-38 shows its contents.



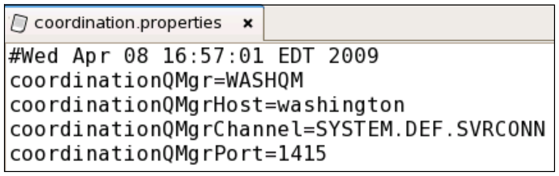
```
#Wed Apr 08 16:57:43 EDT 2009
connectionQMgr=NYQM
```

Figure 3-38 command.properties

3.6.4 coordination.properties

The coordination.properties file specifies the coordination queue manager connection details.

This properties file is located in the /wmbadmin/IBM/WMQFTE/config/WASHQM directory. Figure 3-39 shows its contents.

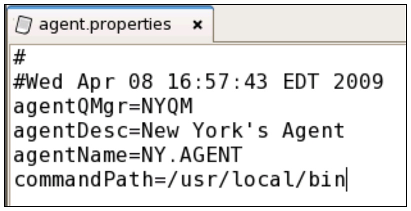


```
#Wed Apr 08 16:57:01 EDT 2009
coordinationQMgr=WASHQM
coordinationQMgrHost=washington
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
coordinationQMgrPort=1415
```

Figure 3-39 coordination.properties

3.6.5 agent.properties

The agent.properties file is associated with a particular agent. In our example, this agent is NY.AGENT, as shown in Figure 3-40.



```
#
#Wed Apr 08 16:57:43 EDT 2009
agentQMgr=NYQM
agentDesc=New York's Agent
agentName=NY.AGENT
commandPath=/usr/local/bin
```

Figure 3-40 Agent properties

There are more than 30 parameters that can be specified for an agent. The agent minimally contains the name of the agent and information about how the agent connects to its queue manager. In Figure 3-40, these are the agentName and agentQMgr parameters. The majority of the possible agent parameters are associated with how the agent runs. These parameters can be important for tuning the agent’s performance.

In Figure 3-40, there is a parameter that is important to discuss, the commandPath parameter. This parameter specifies an authorized path for the agent to call executables. In our scenario, /usr/local/bin is specified because that is the location of the cURL executable that will be used in many of our fteAnt command files.

3.7 Apache Ant

As of FTE release v7.0.1, Ant is automatically installed as part of any FTE installation: FTE Client, FTE Server, or FTE Tools. Figure 3-41 on page 69 shows the ANT directory.

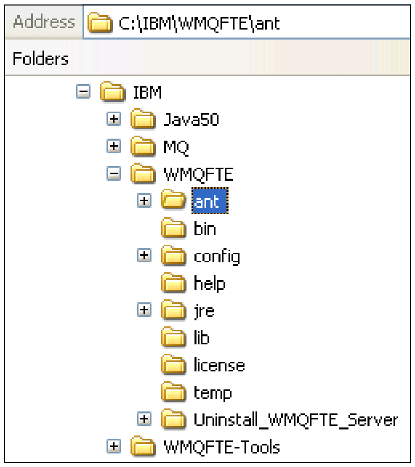


Figure 3-41 Ant directory

Apache Ant provides a solution to many file transfer and integration challenges that are part of this book's business scenario. Figure 3-42 shows that in our scenario, fteAnt is run from the WASHINGTON machine.

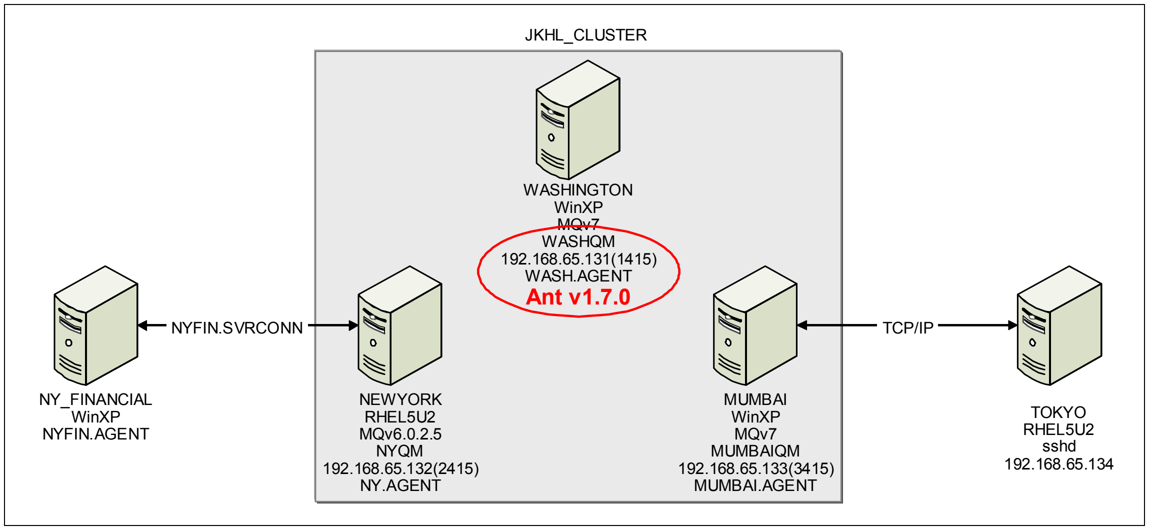


Figure 3-42 In our scenario, fteAnt is run from the WASHINGTON machine

Ant's capabilities are accessed using the **`fteAnt`** command. To confirm that Ant is working after you install FTE, execute the command in Example 3-24 on page 70.

Example 3-24 Confirming that Ant is working

```
C:\IBM\WMQFTE\bin>fteAnt -version
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
FTE Command Build level: f000-20090506-1534
BFGCL0211I: Apache Ant version 1.7.0 compiled on December 13 2006
```

Using fteAnt command: Apache Ant was automatically installed on WASHINGTON, NEWYORK, and MUMBAI FTE Server machines. Therefore, you can use the fteAnt command on all of those machines.

For further information about Apache Ant, visit the following Web page:

http://ant.apache.org/

3.8 cURL

cURL is the Open Source/Free Software command line tool for transferring files with URL syntax. cURL is an essential component in this book's solution scenarios. You must install cURL on the NEWYORK RHELU5 machine, as illustrated in Figure 3-43.

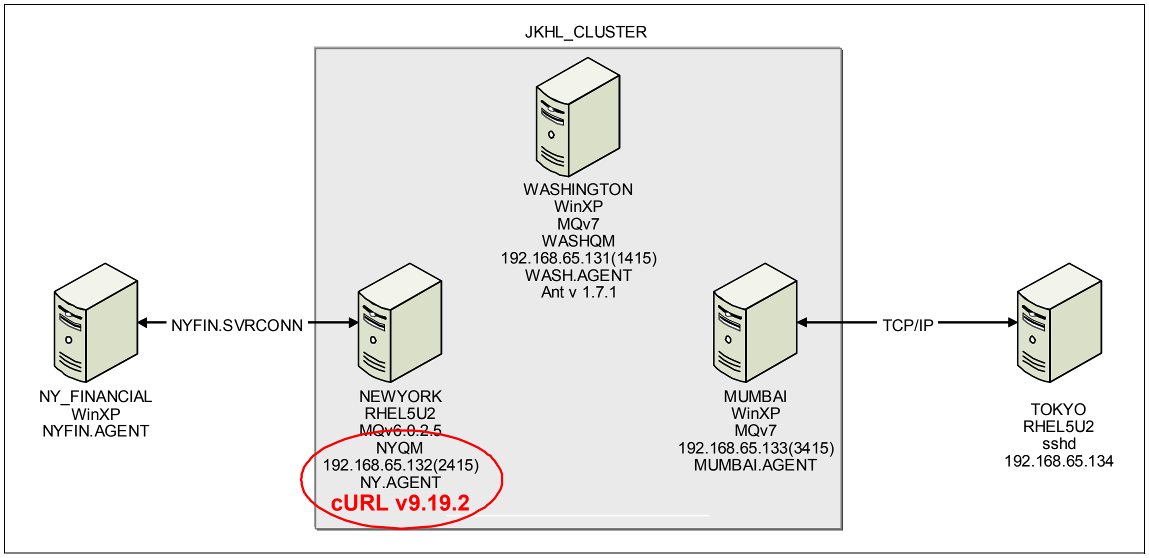


Figure 3-43 Use of cURL on the NEW YORK machine

cURL is used in combination with fteAnt to provide managed file transfer (MFT) access to the TOKYO office through Secure File Transfer Protocol (SFTP).

Precompiled cURL binaries are available from the cURL Web site at:

http://curl.haxx.se/download.html

Available binaries generally do not support the SFTP protocol, as shown on the WASHINGTON machine in Example 3-25.

Example 3-25 No support for the SFTP protocol

```
C:\>curl -V
curl 7.19.2 (i386-pc-win32) libcurl/7.19.2 OpenSSL/0.9.8i zlib/1.2.3
Protocols: tftp ftp telnet dict ldap http file https ftps
Features: Largefile NTLM SSL SSPI libz
```

Note that the required protocol, SFTP, is not listed. To enable cURL SFTP:

1. Download the libssh2 library from

   http://sourceforge.net/projects/libssh2/

2. Install the libssh2 library.
3. Download the cURL source from

   http://curl.haxx.se/download.html

4. Compile and install cURL. This procedure is described at

   http://curl.haxx.se/docs/install.html

At the end of this process, the SFTP protocol will be supported, as shown on the NEWYORK machine in Example 3-26.

Example 3-26 Support for the SFTP protocol

```
[wmbadmin@newyork ~]$ curl -V
curl 7.19.3 (i686-pc-linux-gnu) libcurl/7.19.3 OpenSSL/0.9.8b
zlib/1.2.3 libidn/0.6.5 libssh2/0.17
Protocols: tftp ftp telnet dict http file https ftps scp sftp
Features: IDN IPv6 Largefile NTLM SSL libz
```

For further information about cURL, consult the cURL documentation at:

http://curl.haxx.se/docs/

4

Configuring WebSphere MQ File Transfer Edition on System z

WebSphere MQ File Transfer Edition can run on System z® hardware in two forms:

- On the z/OS operating system
- On the Linux on System z operating system

In this chapter, we describe these two approaches and provide guidelines for z/OS system administrators who want to implement WebSphere MQ File Transfer Edition on System z.

The topics that we discuss in this chapter are:

4.1 WebSphere MQ File Transfer Edition on Linux for System z

Linux on System z was added as a supported operating system in WebSphere MQ File Transfer Edition V7.0.1. WebSphere MQ File Transfer Edition on the Linux for System z platform is not unique from that of any other UNIX® platform. The requirements for this product are the same. The advantage of using this edition of WebSphere MQ File Transfer Edition is that it runs on the System z platform; therefore, it inherits the advantages of the System z environment.

The configuration and administration of WebSphere MQ File Transfer Edition on Linux for System z is identical to distributed platforms. For more information about completing these tasks, refer to:

- Chapter 3, "Installing and configuring WebSphere MQ File Transfer Edition" on page 33
- Chapter 5, "Administering WebSphere MQ File Transfer Edition" on page 89

4.2 WebSphere MQ File Transfer Edition on z/OS

WebSphere MQ File Transfer Edition runs natively on the z/OS platform and executes under UNIX System Services.

WebSphere MQ File Transfer Edition and zAAP

The System z Application Assist Processor (zAAP) specialty engine provides an attractively priced execution environment for Java. It provides Java applications that desire the powerful integration advantages and traditional qualities of service of the IBM mainframe platform.

For Java workloads, zAAPs can maximize the value of their mainframe investments through increased system productivity by reducing the demands and capacity requirements on general purpose processors. zAAPs can lower the overall cost of computing for Java technology-based applications through hardware, software, and maintenance savings.

Because WebSphere MQ File Transfer Edition is written in Java, it is therefore zAAP eligible.

Implementing the JKHL topology on z/OS on System z

Chapter 3 began the construction of the JKHL Travel WebSphere MQ File Transfer Edition environment. The topology we describe is created for a fictional company called JKHL Travel, which we use to demonstrate the product features of WebSphere MQ File Transfer Edition throughout this book. The WebSphere MQ File Transfer Edition for z/OS agent that we create in this chapter is used as the Chicago system in our JKHL topology, as shown in Figure 4-1.

In this section, we provide the following guidance:

- System requirements for WebSphere MQ File Transfer Edition on z/OS
- Installing WebSphere MQ File Transfer Edition for z/OS
- Configuring WebSphere MQ File Transfer Edition for z/OS
- Operational JCL considerations

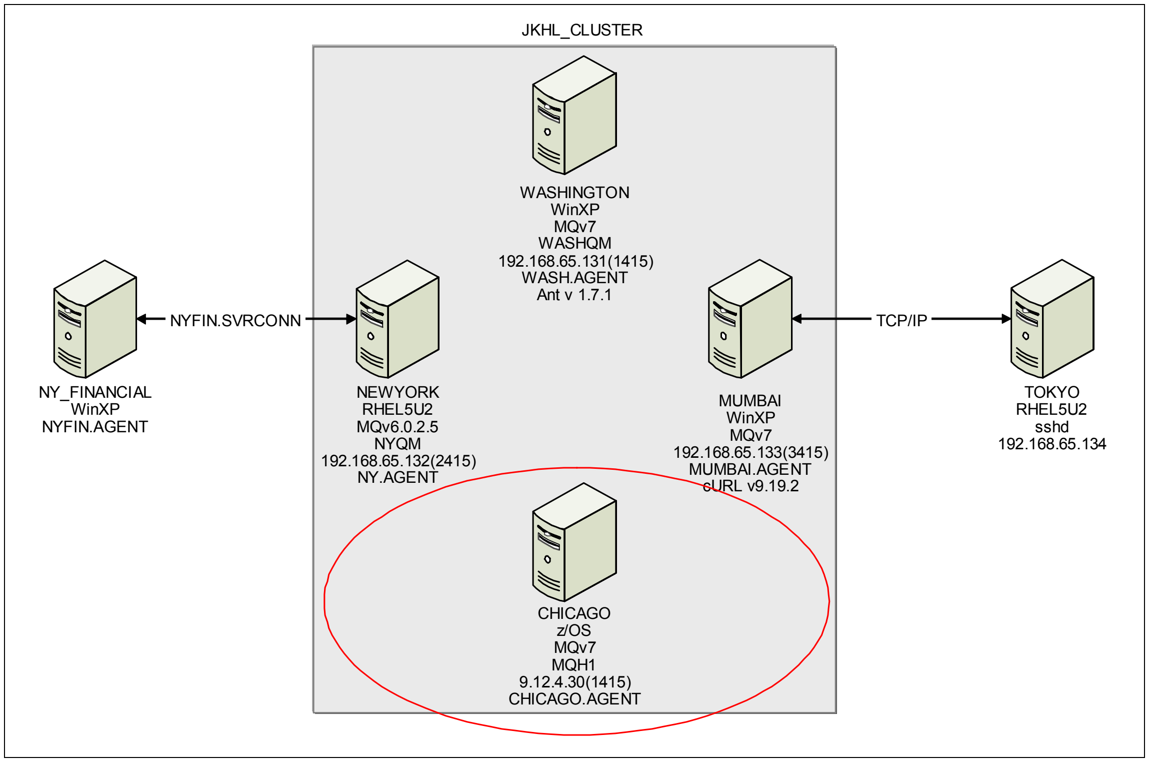


Figure 4-1 Chicago system z/OS in the topology

4.2.1 System requirements for WebSphere MQ File Transfer Edition

The system requirements for WebSphere MQ File Transfer Edition on z/OS are described in the Information Center, so refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.install.doc/zos_prereqs.htm

Our environment used:

- z/OS version 01.09.00, JES2, HBB7740
- z/Architecture® installed and active
- WebSphere MQ for z/OS V7.0 with fix PK74350
- WebSphere MQ File Transfer Edition V7.0.1 for z/OS

4.2.2 Installing WebSphere MQ File Transfer Edition

You install the WebSphere MQ File Transfer Edition code using SMPe. Use *Program Directory for WebSphere MQ File Transfer Edition for z/OS* for guidance. This is often done, as it was in our case, by an SMPe administrator.

The installed code is then a mount point to UNIX System Services where the configuration of the WebSphere MQ File Transfer Edition agents is performed.

4.2.3 Configuring WebSphere MQ File Transfer Edition

The steps to configure an WebSphere MQ File Transfer Edition agent are:

1. Set the environment variables, which we described in “Environment variables for WebSphere MQ File Transfer Edition for z/OS” on page 76.
2. Run the **`fteSetupCoordination`** command according to the instructions that we provide in “Running the fteSetupCoordination command” on page 78.
3. Run the **`fteSetupCommands`** command according to the instructions that we provide in “Running the fteSetupCommands command” on page 80.
4. Run the **`fteCreateAgent`** command according to the instructions that we provide in “Running the fteCreateAgent command” on page 80.
5. Set up WebSphere MQ network communications, which we describe in “Setting up WebSphere MQ network communications” on page 81.
6. Consider “WebSphere MQ File Transfer Edition and WebSphere MQ intercommunication” on page 81.
7. Complete the steps that we provide in “Post installation tasks” on page 83.

Environment variables for WebSphere MQ File Transfer Edition for z/OS

After installation, you must set the environment variables in Table 4-1 on page 77 to the values that are valid for your installation before you run the configuration scripts that WebSphere MQ provides. Table 4-1 on page 77 shows the environment variables and an explanation of the values.

Table 4-1 WebSphere MQ File Transfer Edition Environment variable

Environment variable	Value
FTE_JAVA_HOME	Must point to your Java Version 5.0, SR8a installation
FTE_CONFIG	Must point to a location that you have write access to. This environment variable points to the location that the WebSphere MQ File Transfer Edition for z/OS configuration is written.
STEPLIB	Must include the following WebSphere MQ data sets: ▶ SCSQAUTH ▶ SCSQANLE ▶ SCSQLOAD
LIBPATH	Must include the location of your WebSphere MQ Java libraries in UNIX System Services (for WebSphere MQ Version 7, the default is /mqm/V7R0M0/java/lib).
_BPXK_AUTOCVT	ON

Example 4-1 shows a .profile file with the environment variables for WebSphere MQ File Transfer Edition.

Example 4-1 Environment variables

```
# ======================================
# Below is setup for WMQ FTE
# ======================================
# Set up some basic Java, JMS and FTE paths
  export MQPATH=/usr/lpp/mqm/mqhg/
  export JAVA_HOME=/usr/lpp/java/J5.0
  PATH=$JAVA_HOME/bin:.:$PATH
  PATH=$PATH:$MQPATH/java/lib
  PATH=$PATH:/usr/lpp/mqmfte/V7R0M0/mqhg/bin
  export PATH
  LIBPATH=$MQPATH/java/lib:$LIBPATH:$JAVA_HOME/bin
  LIBPATH=$LIBPATH:$MQPATH/java/lib
  export LIBPATH
  CLASSPATH=.:$MQPATH/java/lib/providerutil.jar
CLASSPATH=$CLASSPATH:$MQPATH/java/lib/com.ibm.mqjms.jar
  CLASSPATH=$CLASSPATH:$MQPATH/java/lib/jms.jar
  CLASSPATH=$CLASSPATH:$MQPATH/java/lib/com.ibm.mq.jar
  CLASSPATH=$CLASSPATH:$MQPATH/java/lib/connector.jar
  export CLASSPATH
# For MQ FTE v7
  export FTE_JAVA_HOME=$JAVA_HOME
  export FTE_CONFIG=/var/mqmfte
  export STEPLIB=MQ700.SCSQAUTH:MQ700.SCSQANLE:MQ700.SCSQLOAD
  export _BPXK_AUTOCVT=ON
  echo Welcome. Your path is: $PATH
#echo
echo path: $PATH
echo libpath: $LIBPATH
echo classpath: $CLASSPATH
echo steplib: $STEPLIB
echo FTE_JAVA_HOME: $FTE_JAVA_HOME
echo FTE_CONFIG: $FTE_CONFIG
echo _BPXK_AUTOCVT: $_BPXK_AUTOCVT
```

You must include the _BPXK_AUTOCVT environment variable in your .profile to interpret the WebSphere MQ File Transfer Edition scripts. However, 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 WebSphere MQ File Transfer Edition scripts, and you can use this script to provide custom environment settings for WebSphere MQ File Transfer Edition.

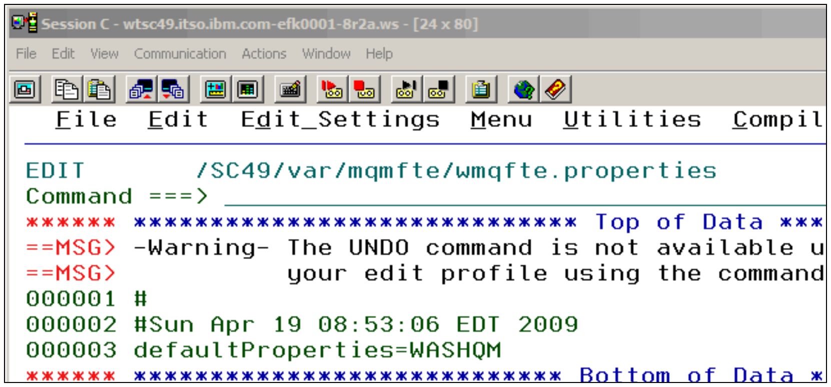
Running the fteSetupCoordination command

The **fteSetupCoordination** command creates the wmqfte.properties file and coordination.properties file for WebSphere MQ File Transfer Edition. It also creates a file that contains the WebSphere MQ definitions for the coordination queue manager. Example 4-2 show the **fteSetupCoordination** command that we used.

Example 4-2 Using the fteSetupCoordination command

```
fteSetupCoordination
      -coordinationQMgr WASHQM
      -default
      -coordinationQMgrHost washington
      -coordinationQMgrPort 1415
      -coordinationQMgrChannel SYSTEM.DEF.SVRCONN
```

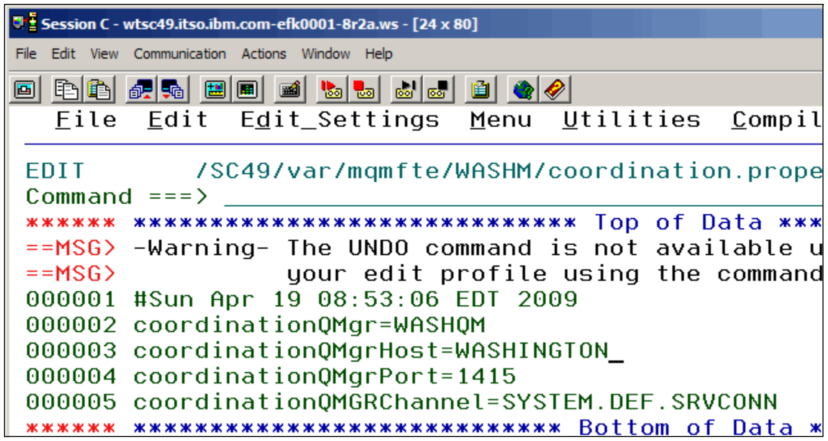
The **fteSetupCoordination** command generates the wmqfte.properties file, as shown in Figure 4-2 on page 79. This file is created in the <FTE configuration data>/ directory.


Session C - wtsc49.itso.ibm.com-efk0001-8r2a.ws - [24 x 80]
File Edit View Communication Actions Window Help

```
   File  Edit  Edit_Settings  Menu  Utilities  Compil

 EDIT        /SC49/var/mqmfte/wmqfte.properties
 Command ===> 
 ****** ****************************** Top of Data ***
 ==MSG> -Warning- The UNDO command is not available u
 ==MSG>           your edit profile using the command
 000001 #
 000002 #Sun Apr 19 08:53:06 EDT 2009
 000003 defaultProperties=WASHQM
 ****** ***************************** Bottom of Data *
```

Figure 4-2   wmqfte.properties file

The **fteSetupCoordination** command also generates the coordination.properties file shown in Figure 4-3. This file is created in the <FTE configuration data>/<coordination queue manager name> directory.


Session C - wtsc49.itso.ibm.com-efk0001-8r2a.ws - [24 x 80]
File Edit View Communication Actions Window Help

```
   File  Edit  Edit_Settings  Menu  Utilities  Compil

 EDIT        /SC49/var/mqmfte/WASHM/coordination.prope
 Command ===> 
 ****** ****************************** Top of Data ***
 ==MSG> -Warning- The UNDO command is not available u
 ==MSG>           your edit profile using the command
 000001 #Sun Apr 19 08:53:06 EDT 2009
 000002 coordinationQMgr=WASHQM
 000003 coordinationQMgrHost=WASHINGTON_
 000004 coordinationQMgrPort=1415
 000005 coordinationQMGRChannel=SYSTEM.DEF.SRVCONN
 ****** ***************************** Bottom of Data *
```

Figure 4-3   Contents of the coordination.properties file

Run the **fteSetupCoordination** command even if you already set up a coordination queue manager on this or another system. If you already updated the coordination queue manager object definitions, you do not need to run the MQSC script generated by the command against the coordination queue manager again.

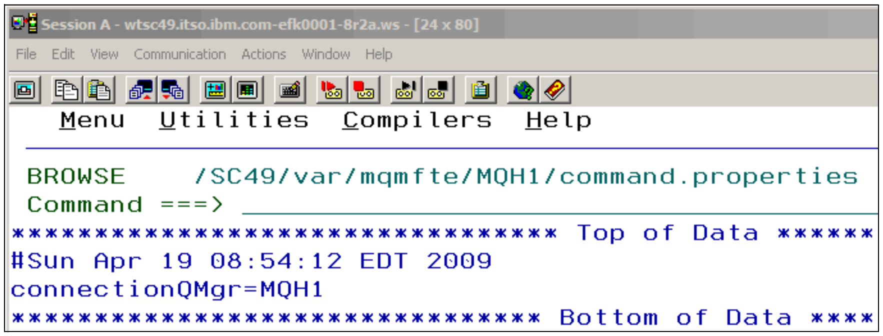
Because we join an WebSphere MQ File Transfer Edition network that already exists, we do not process the definitions. The coordination definition is shown in 3.3.1, “Installing the server on the coordination queue manager” on page 40.

Running the fteSetupCommands command

The **fteSetupCommand** command creates the command.properties file. This properties file specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands. The format of the command line that we use is:

```
fteSetupCommands -connectionQMgr MQH1
```

Notice that we do not provide all of the parameters because we use bindings mode to connect to the z/OS queue manager, which results in the command.properties file shown in Figure 4-4.


Session A - wtsc49.itso.ibm.com-efk0001-8r2a.ws - [24 x 80]
File Edit View Communication Actions Window Help

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------
 BROWSE    /SC49/var/mqmfte/MQH1/command.properties
 Command ===> ________________________________________
********************************* Top of Data ******
#Sun Apr 19 08:54:12 EDT 2009
connectionQMgr=MQH1
******************************** Bottom of Data ****
```

Figure 4-4 Contents of the command.properties file

The command.properties file is created in the <FTE configuration data>/<coordination queue manager name>/agents/<agent name> directory.

For information about all of the parameters for **fteSetupCommands**, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/setup_cmds_cmd.htm

Running the fteCreateAgent command

The fteCreateAgent command creates an agent and its associated configuration. This command gives you the MQSC commands file that you must run against your agent's queue manager to create the following agent queues:

- 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. We show a definition example in 3.3.1, “Installing the server on the coordination queue manager” on page 40.

The agent.properties file is created in the <FTE configuration data>/<coordination queue manager name> directory.

Setting up WebSphere MQ network communications

The z/OS queue manager can join the WebSphere MQ File Transfer Edition network as a participant in a cluster. This union is a leading practice and minimizes the WebSphere MQ administration effort.

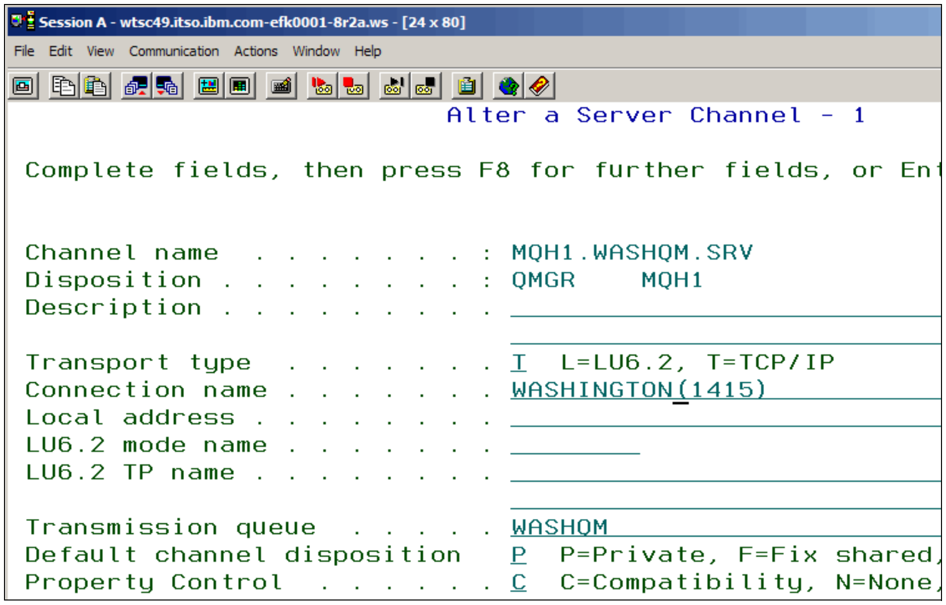
Bindings connections: WebSphere MQ File Transfer Edition agent can use bindings connections to the coordination queue manager only if the agents are on the system as the coordination queue manager.

Certain WebSphere MQ File Transfer Edition tooling commands and the WebSphere MQ Explorer plug-in use managed non-durable subscriptions and therefore must be directly connected to the coordination queue manager. For these, the connection to the coordination queue manager from other systems (even other z/OS systems) must be through client SVRCONNs.

WebSphere MQ File Transfer Edition and WebSphere MQ intercommunication

In our environment, we used message channels to overcome firewall restrictions. We used servers and requesters for our sending channels to each of the other agent queue managers in the network. Figure 4-5 on page 82 shows an example of a server channel.

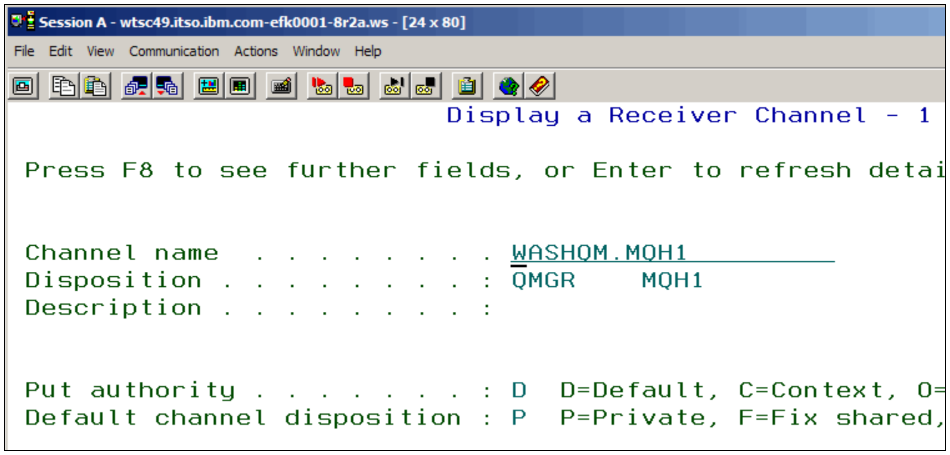
CONVERT parameter: For sender channels it is important that you set the CONVERT parameter to **No**; otherwise, data can become corrupted.


Session A - wtsc49.itso.ibm.com-efk0001-8r2a.ws - [24 x 80]
File Edit View Communication Actions Window Help

```
                           Alter a Server Channel - 1

 Complete fields, then press F8 for further fields, or Ent


 Channel name . . . . . . . . : MQH1.WASHQM.SRV
 Disposition . . . . . . . . : QMGR    MQH1
 Description . . . . . . . . . ____________________________

 Transport type . . . . . . . T  L=LU6.2, T=TCP/IP
 Connection name . . . . . . . WASHINGTON(1415)
 Local address . . . . . . . . ____________________________
 LU6.2 mode name . . . . . . . ________
 LU6.2 TP name . . . . . . . . ____________________________

 Transmission queue . . . . . WASHQM
 Default channel disposition   P  P=Private, F=Fix shared,
 Property Control . . . . . . C  C=Compatibility, N=None,
```

Figure 4-5   Server channel

We used sender and receiver message channels as our receiver channel, and defined a message channel to each of the WebSphere MQ File Transfer Edition agents queue managers. Figure 4-6 shows a receiver channel.


Session A - wtsc49.itso.ibm.com-efk0001-8r2a.ws - [24 x 80]
File Edit View Communication Actions Window Help

```
                           Display a Receiver Channel - 1

 Press F8 to see further fields, or Enter to refresh detai


 Channel name . . . . . . . . . WASHQM.MQH1
 Disposition . . . . . . . . : QMGR    MQH1
 Description . . . . . . . . :


 Put authority . . . . . . . : D  D=Default, C=Context, O=
 Default channel disposition : P  P=Private, F=Fix shared,
```

Figure 4-6   Receiver channel

Post installation tasks

In this section, we outline tasks to complete after you install WebSphere MQ File Transfer Edition.

Running the MQSC scripts commands

On the z/OS system, the MQSC script commands were executed using JCL patterns. We display two methods:

- The first method is to use OCOPY to move the files to the z/OS file structure and point to it in the CSQUTIL program.
- The second is to copy the MQSC script to a member of a PDS and the execute CSQUTIL against that member.

Remember in the case where you already updated the coordination queue manager WebSphere MQ File Transfer Edition WebSphere MQ objects, you do not need to run the MQSC script that the **`fteSetupCoordination`** command generated a second time for the coordination queue manager.

Example 4-3 shows JCL using ocopy for updating the coordination queue manager. Use the TSO OCOPY command to copy data between an MVS data set and the z/OS UNIX file system.

Example 4-3 JCL for using ocopy

```
//TSO      EXEC PGM=IKJEFT01,REGION=2000K,DYNAMNBR=20,TIME=(,10)
//MQSCIN DD PATH='/var/wmqfte/MQH1/MQH1.mqsc',
//   PATHOPTS=ORDONLY
//MQSCOUT  DD RECFM=FB,LRECL=80,DISP=(NEW,PASS),
//   SPACE=(TRK,(1,1))
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
OCOPY INDD(MQSCIN) OUTDD(MQSCOUT) TEXT
//CSQUTIL  EXEC PGM=CSQUTIL,PARM='MQH1'
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=MQM.V700.SCSQANLE,DISP=SHR
//         DD DSN=MQM.V700.SCSQAUTH,DISP=SHR
//CSQUCMD  DD DSN=*.TSO.MQSCOUT,DISP=(OLD,DELETE)
//SYSIN    DD *
COMMAND
```

Example 4-4 on page 84 shows the syntax for processing WebSphere MQ definitions from a PDS member.

Example 4-4 Processing WebSphere MQ definitions

```
//CSQUTIL  EXEC PGM=CSQUTIL,PARM='MQH1'
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=MQ700.SCSQANLE,DISP=SHR
//         DD DSN=MQ700.SCSQAUTH,DISP=SHR
//CSQUCMD  DD DSN=EFK0001.WMQFTE.CNTL(AGENTMQ),DISP=SHR
//SYSIN    DD *
COMMAND
```

Starting the WebSphere MQ File Transfer Edition agent

You can start a WebSphere MQ File Transfer Edition agent by issuing the **`start agent`** command in UNIX System Services, or you can submit JCL to start it.

> **Note:** The user ID that is used to submit the JCL must have the environment variable setup or they must be included within the JCL.

Example 4-5 shows the syntax for using JCL to start an WebSphere MQ File Transfer Edition agent.

Example 4-5 JCL to start a WebSphere MQ File Transfer Edition agent

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD PATH='&FTELOG./stderr-start',
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD PATH='&FTELOG./stdout-start',
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/fteStartAgent -F CHICAGO.AGENT
/*
//
```

The -F shown in Example 4-5 is needed to allow the task to continue to execute rather then have it start and then immediately terminate.

We discuss the **`fteStartAgent`** command in more detail in 5.4.8, “Starting an agent” on page 136.

4.2.4 Operational JCL considerations

You can administer WebSphere MQ File Transfer Edition agents by issuing commands in UNIX System Services or through JCL.

In this section, we provide some examples of using JCL to issue WebSphere MQ File Transfer Edition commands. For more information about issuing commands in UNIX System Services, refer to Chapter 5, “Administering WebSphere MQ File Transfer Edition” on page 89.

fteStopAgent

Example 4-6 shows the sample JCL for **`fteStopAgent`**.

Example 4-6 fteStopAgent

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD  SYSOUT=* PATH='&FTELOG./stderr'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD SYSOUT=* PATH='&FTELOG./stdout'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/fteStopAgent CHICAGO.AGENT
/*
//
```

fteCreateTransfer

Example 4-7 shows a sample JCL for **`fteCreateTransfer`** from z/OS using QSAM.

Example 4-7 fteCreateTransfer from z/OS QSAM

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=* PATH='&FTELOG./stderr-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD SYSOUT=* PATH='&FTELOG./stdout-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/fteCreateTransfer +
 -da WASH.AGENT -dm WASHQM -sa CHICAGO.AGENT -sm MQH1 -t text +
 -df C:/TRANSFERS/EFK0001-COORDMQ.TXT +
"//WMQFTE.COORDMQ.OUTPUT"
/*
//
```

Example 4-8 shows sample JCL for **fteCreateTransfer** from z/OS using a QSAM PDS member.

Example 4-8 fteCreateTransfer from z/OS QSAM PDS member

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=* PATH='&FTELOG./stderr-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD SYSOUT=* PATH='&FTELOG./stdout-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/fteCreateTransfer +
 -da WASH.AGENT -dm WASHQM -sa CHICAGO.AGENT -sm MQH1 -t text +
 -df C:/TRANSFERS/EFK0002-SMALLMEM6.TXT +
"//WMQFTE.DATA(SMALLMEM)"
/*
//
```

Example 4-9 shows sample JCL for **fteCreateTransfer** from z/OS using a PDS member.

Example 4-9 fteCreateTransfer to z/OS PDS member

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=* PATH='&FTELOG./stderr-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD SYSOUT=* PATH='&FTELOG./stdout-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/fteCreateTransfer +
-sa WASH.AGENT -sm WASHQM -da CHICAGO.AGENT -dm MQH1 -t text +
-w -de overwrite -ds "//WMQFTE.DATA(SMALLMEM)" +
C:/TRANSFERS/EFK0002-SMALLMEM7.TXT
/*
//
```

ftePingAgent

Example 4-10 shows sample JCL for **ftePingAgent**.

Example 4-10 ftePingAgent

```
// SET FTELOG='/u/efk0001/ftelog'
//RUN EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=* PATH='&FTELOG./stderr-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDOUT DD SYSOUT=* PATH='&FTELOG./stdout-ping1'
//* PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH /pp/mqmftev7/UK00000/bin/ftePingAgent -m WASHQM +
WASH.AGENT
/*
//
```

5

Administering WebSphere MQ File Transfer Edition

In this chapter, we describe in detail the file transfers and agent management functionality of WebSphere MQ File Transfer Edition. This information will help you understand how to control file transfers and how to administer FTE agents.

The topics that we discuss in this chapter are:

- 5.1, “Administration overview” on page 90
- 5.2, “Creating, scheduling, and triggering file transfers” on page 99
- 5.3, “Administering file transfers” on page 124
- 5.4, “Administering WebSphere MQ File Transfer Edition agents” on page 129
- 5.5, “Extended management in WebSphere MQ File Transfer Edition” on page 140

In this chapter, we do not cover WebSphere MQ administration related to these topics only WebSphere MQ File Transfer Edition administration. As you go through this chapter, keep in mind that more information about these topics and settings as they pertain to WebSphere MQ are available n the WebSphere MQ information center, which is located at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

5.1 Administration overview

At first glance, there are four methods to trigger and manage file transfers in WebSphere MQ File Transfer Edition:

- Command line is the most important management tool. It can provide file transfer operation and agent management.
- The fact that WebSphere MQ File Transfer Edition provides an easy-to-use Eclipse plug-in GUI makes it fairly simple to create file transfers and transfer templates.
- The third triggering file transfer method is using file transfer request XML messages. When the standard XML request messages are put to the command queue on the source FTE agent, the file transfers are submitted and run. File transfer request XML messages can be integrated into business applications.
- A resource monitor can also start a file transfer and invoke some actions commands, executable programs, Ant scripts, and Job Control Language (JCL) when the trigger condition is detected.

5.1.1 Administration with commands

The command line is an important management tool of WebSphere MQ File Transfer Edition. All FTE objects can be administered with the command line. In Table 5-1, we describe every command name and its purpose.

Table 5-1 WebSphere MQ File Transfer Edition commands and their purpose

Command name	Purpose
Commands for configuration:	
fteCreateAgent	Creates a new WebSphere MQ File Transfer Edition agent
fteDeleteAgent	Deletes a particular WebSphere MQ File Transfer Edition agent
fteChangeDefaultConfigurationOptions	Changes the default configuration options that you want WebSphere MQ File Transfer Edition to use
fteSetupCoordination	Configures a WebSphere MQ File Transfer Edition coordination queue manager
fteSetupCommands	Specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands
fteCreateMonitor	Creates and starts a new resource monitor using a WebSphere MQ File Transfer Edition agent

Command name	Purpose
fteDeleteMonitor	Deletes an existing WebSphere MQ File Transfer Edition resource monitor
fteListMonitors	Lists all of the existing resource monitors
Commands for administration:	
fteStartAgent	Starts a particular agent before using it to transfer files
fteCreateTransfer	Creates a new file transfer for immediate or scheduled start
fteListAgents	Lists all of the agents that are registered against a particular coordination queue manager
fteShowAgentDetails	Displays the details of a particular agent
fteListScheduledTransfers	Lists all of the scheduled file transfers that you previously scheduled for a source agent
fteDeleteScheduledTransfer	Deletes a particular file transfer that you previously scheduled
fteCleanAgent	Cleans up an agent's queues
fteStopAgent	Stops a particular agent
fteCancelTransfer	Cancels a file transfer
fteAnt	Allows you to integrate file transfer function into the Apache Ant tool and use file transfer Ant tasks from your Ant scripts to coordinate complex file transfer operations from an interpreted scripting language
ftePingAgent	Pings an agent to determine whether the agent is active and able to process transfers
Command for troubleshooting:	
fteSetAgentTraceLevel	Sets the level of agent trace to run

Depending on the command's purpose, you can use WebSphere MQ File Transfer Edition commands for either local administration, remote administration, or both, for example, you can use commands, such as fteCreateAgent and fteStartAgent, to perform actions on the system where you issue the command only. You cannot create or start an agent on a remote system. You can run other commands, such as fteShowAgentDetails and fteStopAgent, remotely provided that you have access to the remote system.

Table 5-2 summarizes the location where the command can run. If there is X character for a command, this means that the command supports this mode.

Table 5-2 Commands issued locally and remotely

Name	Local commands	Remote commands
fteCreateTransfer	X	X
fteCancelTransfer	X	X
fteListScheduledTransfers	X	X
fteDeleteScheduledTransfer	X	X
fteListAgents	X	X
fteShowAgentDetails	X	X
fteSetAgentTraceLevel	X	X
fteCreateAgent	X	
fteStartAgent	X	
fteStopAgent	X	X
fteCleanAgent	X	
fteDeleteAgent	X	
fteSetupCommands	X	
fteSetupCoordination	X	
fteChangeDefaultConfiguration	X	
fteCreateMonitor	X	
fteDeleteMonitor	X	X
fteListMonitors	X	X
fteAnt	X	X
ftePingAgent	X	X

The remote command set contains commands to start and manage file transfers:

- fteCreateTransfer
- fteCancelTransfer
- fteListScheduledTransfers
- fteDeleteScheduledTransfer.

There are remote commands for looking at what agents are available:

- fteListAgents

- fteShowAgentDetails

There is also the ability to turn on trace of an agent remotely:

- fteSetAgentTraceLevel

What is not included in the remote commands set are commands to create, manage, and delete agents. The local commands for managing agents are:

- fteCreateAgent
- fteStartAgent
- fteCleanAgent
- fteDeleteAgent

We must pay attention to the **`fteStopAgent`** command. In fact, it is also a remote command. You can run the command on any FTE agent hosted system that connects to a WebSphere MQ FTE network.

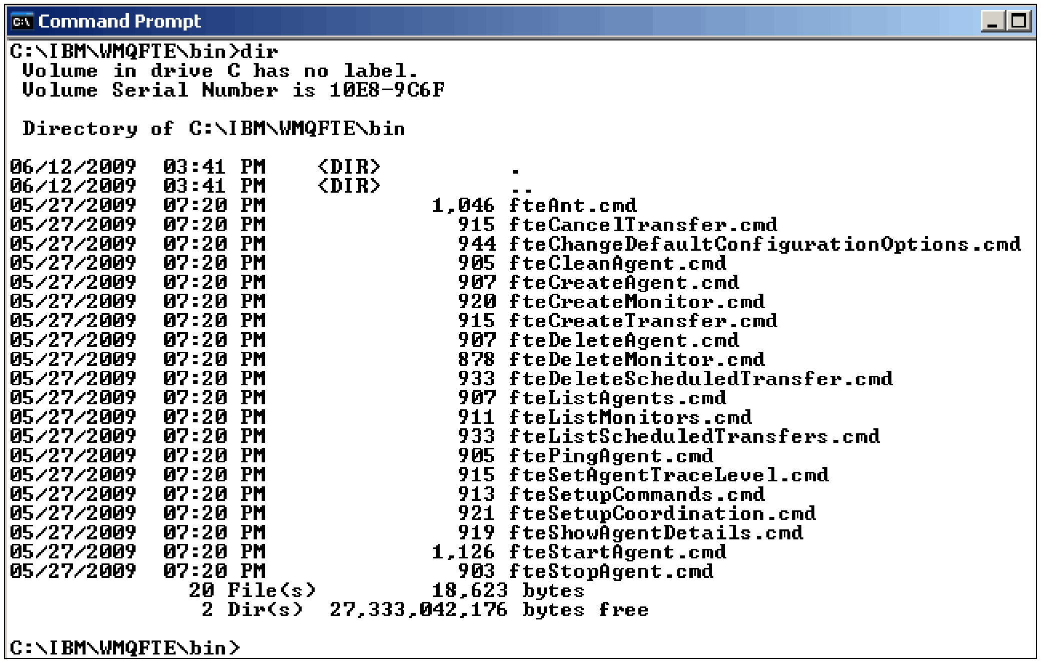
Finally, the commands to set up the configuration for FTE agent and remote tools are also local administration type and they include:

- fteSetupCommands
- fteSetupCoordination
- fteChangeDefaultConfiguration.

In the next sections, we discuss the command interfaces on different platforms.

WebSphere MQ File Transfer Edition for Windows

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. Figure 5-1 on page 94 shows the command directory of a server installation WebSphere MQ File Transfer Edition on Windows.

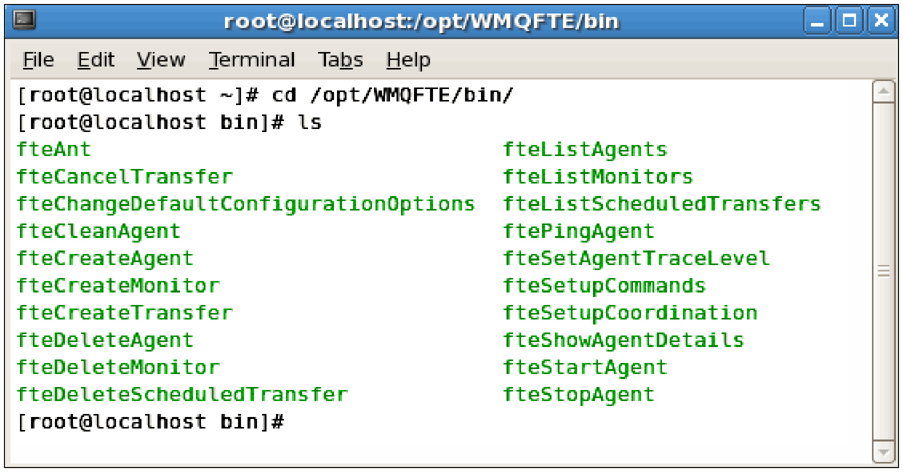


```
C:\IBM\WMQFTE\bin>dir
 Volume in drive C has no label.
 Volume Serial Number is 10E8-9C6F

 Directory of C:\IBM\WMQFTE\bin

06/12/2009  03:41 PM    <DIR>          .
06/12/2009  03:41 PM    <DIR>          ..
05/27/2009  07:20 PM             1,046 fteAnt.cmd
05/27/2009  07:20 PM               915 fteCancelTransfer.cmd
05/27/2009  07:20 PM               944 fteChangeDefaultConfigurationOptions.cmd
05/27/2009  07:20 PM               905 fteCleanAgent.cmd
05/27/2009  07:20 PM               907 fteCreateAgent.cmd
05/27/2009  07:20 PM               920 fteCreateMonitor.cmd
05/27/2009  07:20 PM               915 fteCreateTransfer.cmd
05/27/2009  07:20 PM               907 fteDeleteAgent.cmd
05/27/2009  07:20 PM               878 fteDeleteMonitor.cmd
05/27/2009  07:20 PM               933 fteDeleteScheduledTransfer.cmd
05/27/2009  07:20 PM               907 fteListAgents.cmd
05/27/2009  07:20 PM               911 fteListMonitors.cmd
05/27/2009  07:20 PM               933 fteListScheduledTransfers.cmd
05/27/2009  07:20 PM               905 ftePingAgent.cmd
05/27/2009  07:20 PM               915 fteSetAgentTraceLevel.cmd
05/27/2009  07:20 PM               913 fteSetupCommands.cmd
05/27/2009  07:20 PM               921 fteSetupCoordination.cmd
05/27/2009  07:20 PM               919 fteShowAgentDetails.cmd
05/27/2009  07:20 PM             1,126 fteStartAgent.cmd
05/27/2009  07:20 PM               903 fteStopAgent.cmd
              20 File(s)         18,623 bytes
               2 Dir(s)  27,333,042,176 bytes free

C:\IBM\WMQFTE\bin>
```

Figure 5-1 The command line of WebSphere MQ File Transfer Edition is shown on Windows

WebSphere MQ File Transfer Edition for UNIX and Linux systems

You can issue all WebSphere MQ File Transfer Edition commands from a shell on UNIX and Linux. All commands are case-sensitive. Figure 5-2 shows the commands of WebSphere MQ File Transfer Edition on Linux.



```
[root@localhost ~]# cd /opt/WMQFTE/bin/
[root@localhost bin]# ls
fteAnt                                fteListAgents
fteCancelTransfer                     fteListMonitors
fteChangeDefaultConfigurationOptions  fteListScheduledTransfers
fteCleanAgent                         ftePingAgent
fteCreateAgent                        fteSetAgentTraceLevel
fteCreateMonitor                      fteSetupCommands
fteCreateTransfer                     fteSetupCoordination
fteDeleteAgent                        fteShowAgentDetails
fteDeleteMonitor                      fteStartAgent
fteDeleteScheduledTransfer            fteStopAgent
[root@localhost bin]#
```

Figure 5-2 The WebSphere MQ File Transfer Edition commands are shown on Linux

WebSphere MQ File Transfer Edition for z/OS

You can invoke WebSphere MQ File Transfer Edition commands from JCL for integration into batch suites on z/OS.

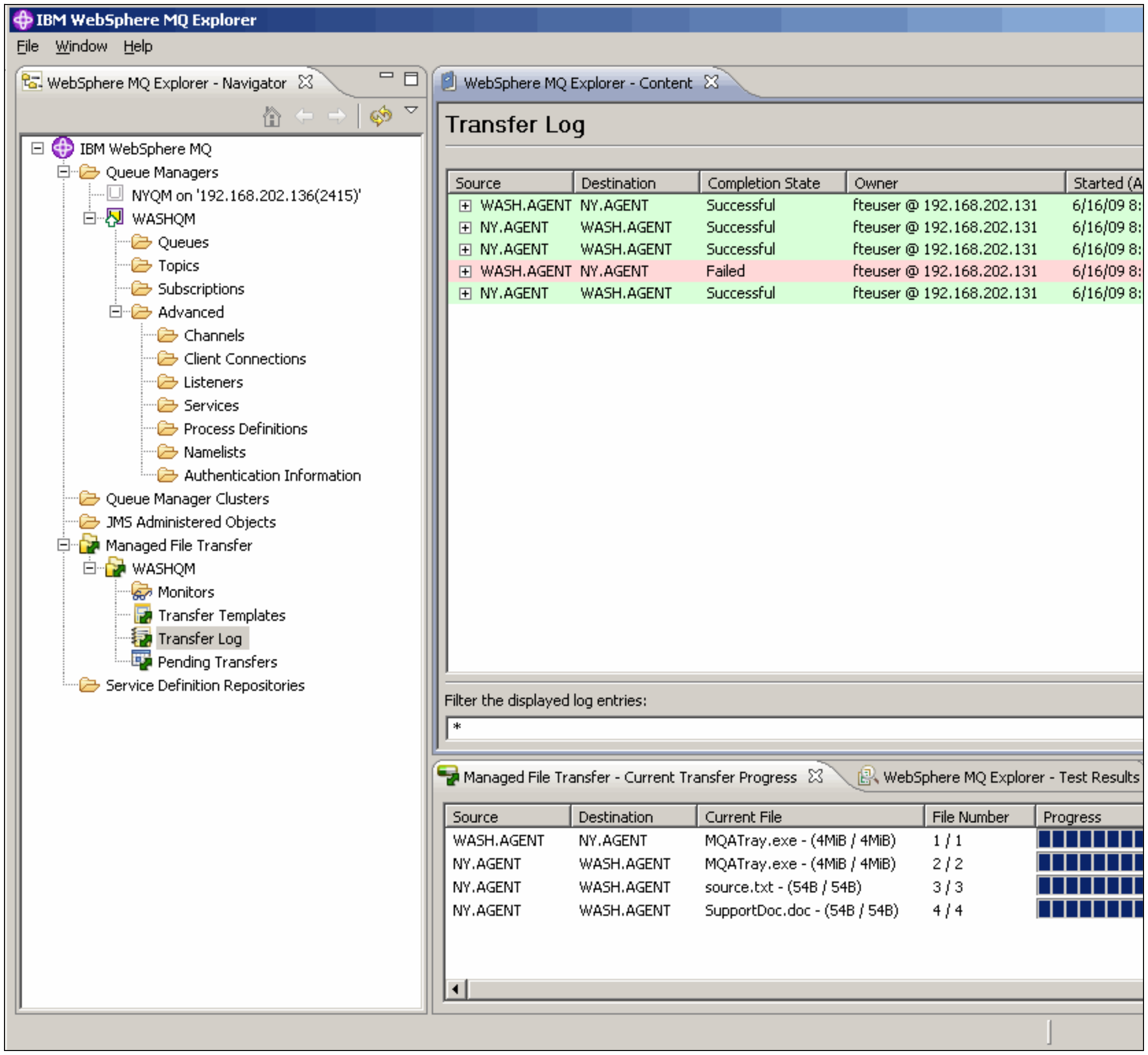
For more information about submitting z/OS commands, refer to Chapter 4, “Configuring WebSphere MQ File Transfer Edition on System z” on page 73.

5.1.2 Managing file transfers in WebSphere MQ Explorer

Using the extension (plug-in) for the WebSphere MQ Explorer that WebSphere MQ File Transfer Edition includes, you can easily administer file transfers. The administration tasks that you can run in the FTE Eclipse plug-in GUI are:

- Creating and cancelling file transfers
- Creating transfer templates
- Creating and deleting resource monitors
- Viewing and deleting scheduled transfers
- Monitoring the real-time status of file transfers

Figure 5-3 on page 96 shows the WebSphere MQ File Transfer Edition plug-in for WebSphere MQ Explorer.



Transfer Log

Source	Destination	Completion State	Owner	Started (A
⊞ WASH.AGENT	NY.AGENT	Successful	fteuser @ 192.168.202.131	6/16/09 8:
⊞ NY.AGENT	WASH.AGENT	Successful	fteuser @ 192.168.202.131	6/16/09 8:
⊞ NY.AGENT	WASH.AGENT	Successful	fteuser @ 192.168.202.131	6/16/09 8:
⊞ WASH.AGENT	NY.AGENT	Failed	fteuser @ 192.168.202.131	6/16/09 8:
⊞ NY.AGENT	WASH.AGENT	Successful	fteuser @ 192.168.202.131	6/16/09 8:

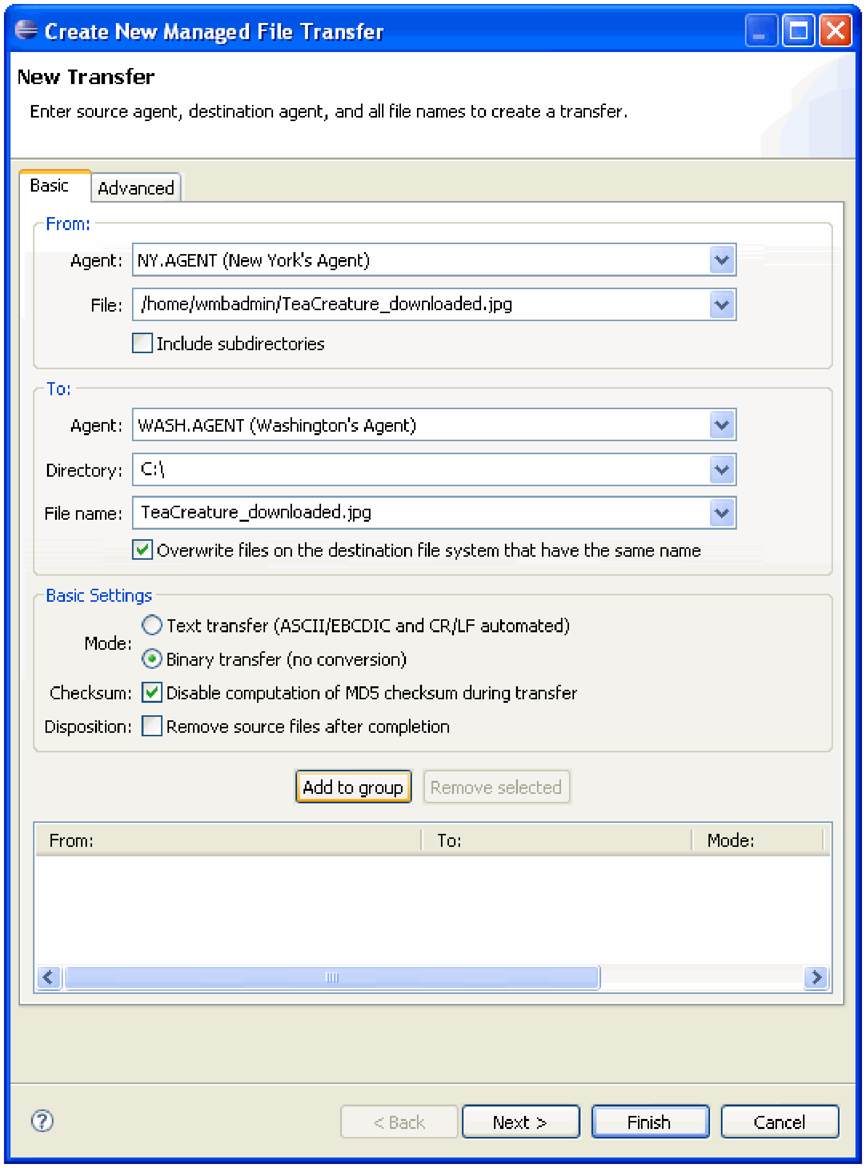
Filter the displayed log entries:

```
*
```

Source	Destination	Current File	File Number	Progress
WASH.AGENT	NY.AGENT	MQATray.exe - (4MiB / 4MiB)	1 / 1	■■■■■■■■
NY.AGENT	WASH.AGENT	MQATray.exe - (4MiB / 4MiB)	2 / 2	■■■■■■■■
NY.AGENT	WASH.AGENT	source.txt - (54B / 54B)	3 / 3	■■■■■■■■
NY.AGENT	WASH.AGENT	SupportDoc.doc - (54B / 54B)	4 / 4	■■■■■■■■

Figure 5-3 WebSphere MQ File Transfer Edition plug-in for WebSphere MQ Explorer

You can open a new managed file transfer window and select file transfer options on the WebSphere MQ File Transfer Edition Eclipse GUI, and then submit the transfer, for instance, in Figure 5-4 on page 97, you can choose the source agent and the destination agent from the agent lists, input the file or directory names on the source and destination agents, select the option of overwriting files on the destination file systems that have the same names, and set the appropriate priority. You can also specify multiple source and destination pairs in the MQ Explorer plug-in by clicking **Add to group**.


Create New Managed File Transfer
New Transfer
Enter source agent, destination agent, and all file names to create a transfer.
Basic
Advanced

From:

Agent: NY.AGENT (New York's Agent)

File: /home/wmbadmin/TeaCreature_downloaded.jpg

[ ] Include subdirectories

To:

Agent: WASH.AGENT (Washington's Agent)

Directory: C:\

File name: TeaCreature_downloaded.jpg

[✓] Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ( ) Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ Binary transfer (no conversion)]

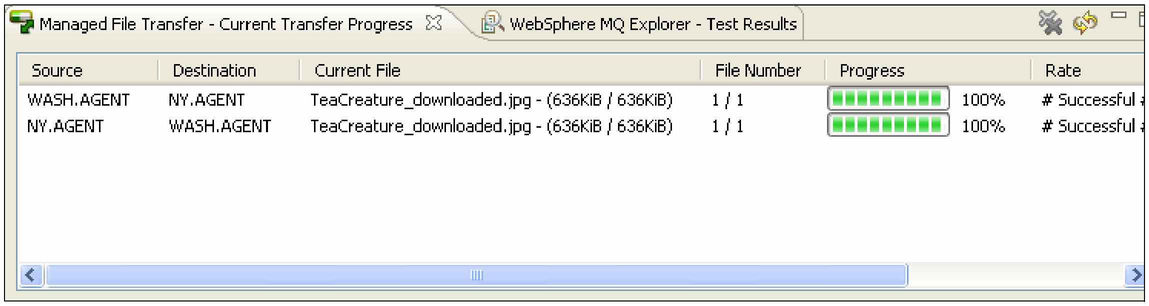
Checksum: [✓] Disable computation of MD5 checksum during transfer

Disposition: [ ] Remove source files after completion

From:	To:	Mode:

Figure 5-4 The window of creating new managed file transfer

It is important that the plug-in GUI can also monitor the real-time file transfer state, view the transfer logs, and schedule information of file transfers. Figure 5-5 on page 98 shows the real-time file transfer status.


Managed File Transfer - Current Transfer Progress
WebSphere MQ Explorer - Test Results

Source	Destination	Current File	File Number	Progress	Rate
WASH.AGENT	NY.AGENT	TeaCreature_downloaded.jpg - (636KiB / 636KiB)	1 / 1	100%	# Successful
NY.AGENT	WASH.AGENT	TeaCreature_downloaded.jpg - (636KiB / 636KiB)	1 / 1	100%	# Successful

Figure 5-5 The real-time file transfer status window

5.1.3 File transfer XML request messages

XML messages that arrive at an agent's command queue initiate File transfers, typically as a result of a user issuing a file transfer command or using the WebSphere MQ Explorer interface. File transfer messages can have one of three root elements:

- <request>: For new file transfer requests or deleting scheduled transfers that are pending
- <cancel>: For canceling file transfers in progress
- <transferSpecifications>: For specifying multiple transfer file groups that the **`fteCreateTransfer`** command uses

FileTransfer.xsd schema document: The FileTransfer.xsd schema document is located in the MessageSchemas directory on the Remote Tools and Documentation CD. The FileTransfer.xsd schema imports fteutils.xsd, which is at the same location on the CD.

Example 5-1 shows the format of a simple file transfer request XML message that is transferring a single file to a different agent.

Example 5-1 Simple example of file transfer request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>washington.jkhl.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="WASH.AGENT" QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT" QMgr="NYQM"/>
    <transferSet priority="0">
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\source1.doc</file>
        </source>
        <destination type="file" exist="error">
          <file>/home/wmbadmin/dest2.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

5.2 Creating, scheduling, and triggering file transfers

In this section, we describe how to create new managed file transfers, which are started by command line, FTE Eclipse plug-in GUI, and file transfer request messages. In this section, we also provide information about setting scheduled transfers and triggered transfers and using a resource monitor to create and start a file transfer when the condition is satisfied.

5.2.1 Creating new managed file transfers

You can start a new file transfer from the WebSphere MQ Explorer or from the command line, and you can choose to transfer either a single file or multiple files in a group. You can also initiate a file transfer by putting an XML request message on the source agent's command queue. You can specify multiple source files for a file transfer in multiple destination locations, if all destinations are on the same agent.

Starting a new file transfer using the command line

The **`fteCreateTransfer`** command creates and starts a file transfer. Using this command you can start a file transfer immediately and schedule a file transfer for a future time and date to repeat a scheduled transfer one or more times and to trigger a file transfer based on certain conditions.

You can run the **`fteCreateTransfer`** command from any system that can connect to the WebSphere MQ network and subsequently route to the agent's queue manager.

This command uses the command.properties file to connect to the WebSphere 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.

You can specify multiple source files for a file transfer, but you can specify only one destination. The destination list is not supported, which means transferring one file to multiple destinations.

Example 5-2 shows the **fteCreateTransfer** command syntax.

Example 5-2 The fteCreateTransfer command syntax

```
fteCreateTransfer [-p ConfigurationOptions] -sa AgentName
                  [-sm QueueManager] -da AgentName
                  [-dm QueueManager] [-td TransferDefinitionFile] |
                  ([-df File] | [-dd Directory] |
                  [-ds SequentialDataset] | [-dp PartitionedDataset])
                  [-r] [-w] [-t TransferMode] [-cs ChecksumMethod]
                  [-de <actions>] [-sd <disposition>] [-pr <priority>]
                  [-md metaData] ([-tr triggerSpec ] [-tl ])
                  ([-ss startSchedule] [-tb timeBase]
                  [-oi occurrenceInterval] [-of occurrenceFrequency]
                  [-oc occurrenceCount] | [-es endSchedule])
                  -gt [transferTemplateFilePath] [-jn jobName]
                  SourceFileSpec
```

The following list provides an explanation of the syntax in Example 5-2:

- -sa AgentName

  Required. The name of the agent that the source file is transferred from.

- -sm QueueManager

  Optional. The name of the queue manager that the source agent is connected to.

- -da AgentName

  Required. The name of the agent that the file is transferred to.

- -dm QueueManager

  Optional. The name of the queue manager that the destination agent is connected to.

- -td TransferDefinitionFile

  Optional. The name of the transfer definition XML document that defines one or more source and destination file specifications for the transfer.

- SourceFileSpec

  Required if you specified one of the -df, -dd, -dp, or -ds parameters. If you specify the -td parameter, do not specify source_file_specification.

  Source file specifications can include an asterisk as a wildcard that matches zero or more characters. But each file specification must be in one of the following categories:

  - File names
  - Directories
  - Sequential data set
  - Partitioned data set

We introduced basic parameters of the **`fteCreateTransfer`** command. If you want to know more information about all of the parameters for **`fteCreateTransfe`**r, refer to:

`http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/start_new_transfer_cmd.htm`

In this section, we show some examples and explain how to issue this command:

- Basic example

  In this basic example, the file transferPicture.jpg is transferred from WASH.AGENT on Windows to NY.AGENT on Linux and renamed to ny_transferPicture.jpg. Example 5-3 shows the command that we used.

  Example 5-3 The basic transfer example

  ```
  fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM
  -df /home/wmbadmin/ny_transferPicture.jpg C:\transferPicture.jpg
  ```

- Using the options of the priority, overwrite, and non-checksum on file transfer

  In Example 5-4 the file transferPicture.jpg is transferred from WASHQM's system to NYQM's system. The option of priority is set to 5. The destination behavior is overwriting the existing file. The non-checksum parameter is set.

  Example 5-4 Advanced transfer example

  ```
  fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM
  -pr 5 -de overwrite -cs none -df /home/wmbadmin/transferPicture.jpg
  C:\transferPicture.jpg
  ```

- Transferring from distributed platform to z/OS

  In Example 5-5, the file transferText.txt is transferred from WASHQM's system to a data set //'USERID.TRANS.FILE.TXT' on CHICAGOQM's system. Text mode was selected to convert data from ASCII to EBCDIC.

  Example 5-5   File transfer from distributed platform to zOS

```
fteCreateTransfer -t text -sa WASH.AGENT -sm WASHQM
-da CHICAGO.AGENT -dm MQH1 -ds
"//TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);SPACE(5,1)"
C:\transferText.txt
```

- Using transfer definition XML files

  In Example 5-6, you can use an XML editor that supports schema validation to create the example.xml file. In this transfer definition XML file, you can set different options for each file pair, and you can specify different source and destination locations using the same destination agent.

  Example 5-6   Transfer definition file - example.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">

  <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">
```

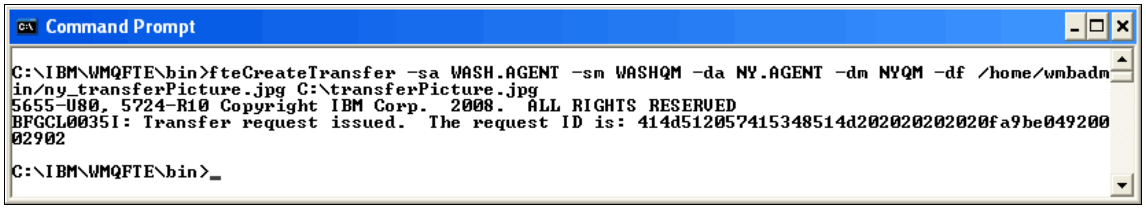
```
    <source recursive="false" disposition="leave">
      <file>c:\originfiles\source3.txt</file>
    </source>
    <destination type="file" exist="overwrite">
      <file>c:\targetfiles\destination3.txt</file>
    </destination>
  </item>
</transferSpecifications>
```

Save the transfer definition file, example.xml, and use the file for a transfer issued from a command line, as shown in Example 5-7.

Example 5-7 Issuing fteCreateTransfer command with transfer definition file

```
fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM
-td C:\example.xml
```

If you submit the **`fteCreateTransfer`** command without the -w parameter, you can get the transfer request ID, as shown in Figure 5-6. The request ID can be used in the **`fteCancelTransfer`** command or to identify the transfer in log messages.



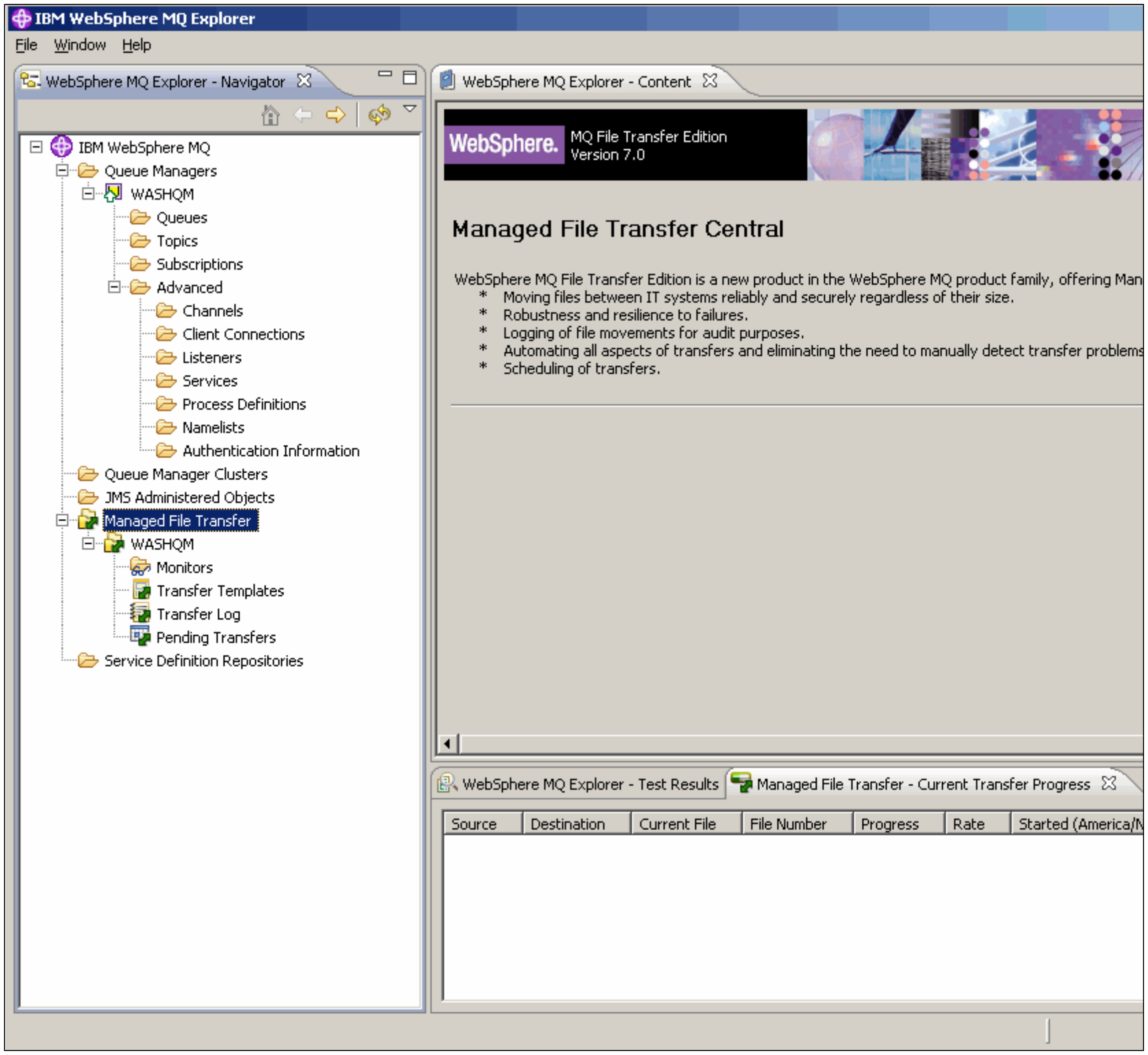
```
C:\IBM\WMQFTE\bin>fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM -df /home/wmbadm
in/ny_transferPicture.jpg C:\transferPicture.jpg
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0035I: Transfer request issued.  The request ID is: 414d512057415348514d202020202020fa9be049200
02902

C:\IBM\WMQFTE\bin>_
```

Figure 5-6 Submitting the transfer request successfully

Creating a new file transfer using the WebSphere MQ Explorer

Use the following steps to start a new file transfer using the Create New Managed File Transfer wizard in the WebSphere MQ Explorer:

1. In the Navigator view in WebSphere MQ Explorer, click **Managed File Transfer**. Managed File Transfer Central is displayed in the Content view, as shown in Figure 5-7 on page 104.



Managed File Transfer Central

WebSphere MQ File Transfer Edition is a new product in the WebSphere MQ product family, offering Man

* Moving files between IT systems reliably and securely regardless of their size.
* Robustness and resilience to failures.
* Logging of file movements for audit purposes.
* Automating all aspects of transfers and eliminating the need to manually detect transfer problems
* Scheduling of transfers.

Figure 5-7 Managed File Transfer GUI on WebSphere MQ Explorer

2. The coordination queue manager is 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 with. If you are currently connected to a coordination queue manager other than the one you want to use for the transfer:

   a. In the Navigator view right-click that coordination queue manager name, and click **Disconnect**.

b. Right-click the name of the coordination queue manager that you want to use, and click **Connect**, as shown in Figure 5-8.

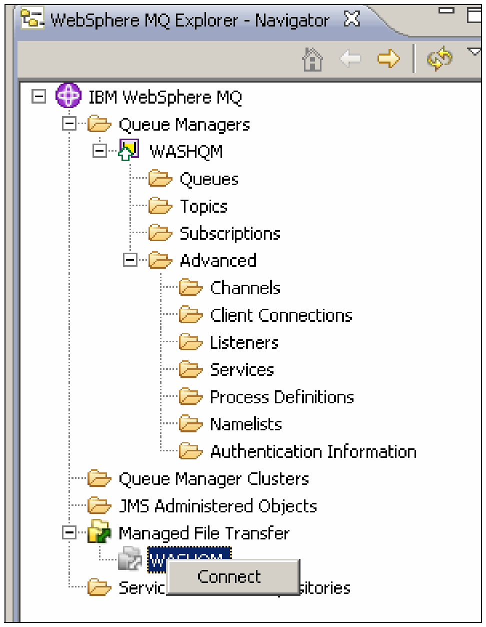


Figure 5-8 Connecting to coordination queue manager

3. Start the Create New Managed File Transfer wizard using either of the following methods:

   – 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. Click **New Transfer** to launch the wizard.

   – Select **IBM WebSphere MQ** → **Managed File Transfer** → **New Transfer Wizard**. See Figure 5-9 on page 106.

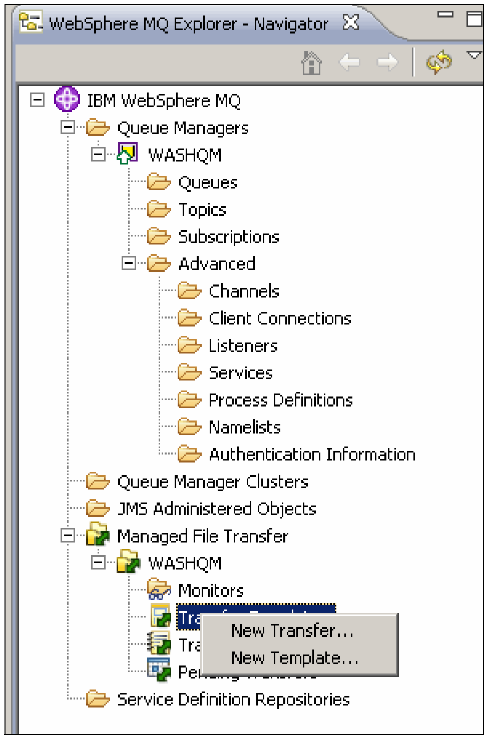
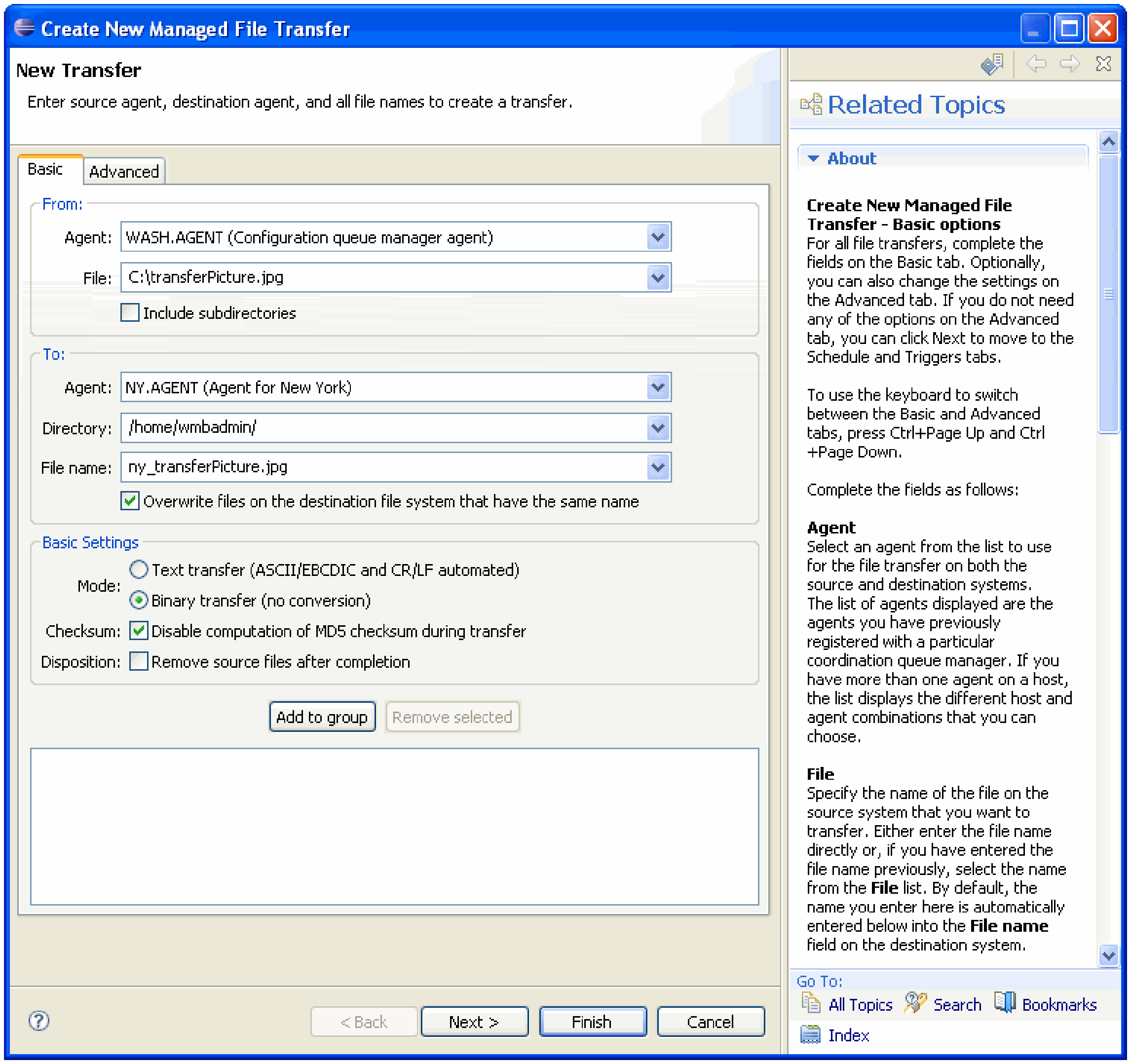


Figure 5-9 Opening a new transfer wizard

4. Follow the instructions on the wizard panels, as shown in Figure 5-10 on page 107. There is also context-sensitive help provided for each panel. To access the context-sensitive help on:
   - Windows, press F1
   - Linux, press Ctrl+F1 or Shift+F1


Create New Managed File Transfer

New Transfer

Enter source agent, destination agent, and all file names to create a transfer.

Basic | Advanced

From:

Agent: WASH.AGENT (Configuration queue manager agent)

File: C:\transferPicture.jpg

☐ Include subdirectories

To:

Agent: NY.AGENT (Agent for New York)

Directory: /home/wmbadmin/

File name: ny_transferPicture.jpg

☑ Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ○ Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ ● Binary transfer (no conversion)]

Checksum: ☑ Disable computation of MD5 checksum during transfer

Disposition: ☐ Remove source files after completion

Related Topics

About

Create New Managed File Transfer - Basic options
For all file transfers, complete the fields on the Basic tab. Optionally, you can also change the settings on the Advanced tab. If you do not need any of the options on the Advanced tab, you can click Next to move to the Schedule and Triggers tabs.

To use the keyboard to switch between the Basic and Advanced tabs, press Ctrl+Page Up and Ctrl +Page Down.

Complete the fields as follows:

Agent
Select an agent from the list to use for the file transfer on both the source and destination systems. The list of agents displayed are the agents you have previously registered with a particular coordination queue manager. If you have more than one agent on a host, the list displays the different host and agent combinations that you can choose.

File
Specify the name of the file on the source system that you want to transfer. Either enter the file name directly or, if you have entered the file name previously, select the name from the **File** list. By default, the name you enter here is automatically entered below into the **File name** field on the destination system.

Figure 5-10 Create New Managed File Transfer wizard

Using a file transfer request message

You can submit a file transfer XML request message into the agent's command queue (SYSTEM.FTE.COMMAND.<AgentName>). The file transfer request message that is generated from the FileTransfer.xsd schema document specifies the details of the transfer, which includes information, such as required user, source and destination agents, source and destination file selection, and transfer behavior. You can use optional information to specify scheduled transfers, trigger conditions, and user-defined metadata.

Example 5-8 shows the format of a simple file transfer XML request message.

Example 5-8   A simple file transfer request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>washington.jkhl.com</hostName>
        <userID>wmbadmin</userID>
    </originator>
    <sourceAgent agent="WASH.AGENT" QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT" QMgr="NYQM"/>
    <transferSet priority="0">
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>C:\transferPicture.jpg</file>
        </source>
        <destination type="file" exist="error">
          <file>/home/wmbadmin/ny_transferPicture.jpg</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

If you want to know more information about XML elements and attributes that are used in transfer request messages, you can visit the *File transfer request message format* section in the WebSphere MQ File Transfer Edition Information Center:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/transfer_message_format.htm

You can implement the function that submits XML request messages into agent's command queues in standard C, .NET, or Java applications that use WebSphere MQ programming interfaces or in standard Java Message Service applications. You can get WebSphere MQ programming guides from the IBM WebSphere MQ Information Center:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

5.2.2 Scheduling file transfers

A scheduled file transfer is a simple, group or wildcard transfer that is executed at a specific time or date. It can be configured to perform the same transfer on a repetitive basis.

After the created or scheduled file transfers, they are saved in the pending transfers. These details are not visible or editable. If you want to change any information in the scheduling transfer that is submitted in the WebSphere MQ Explorer, you must cancel and re-enter them.

Creating the scheduling transfer with the command line

The six parameters of -ss, -tb, -oi, -of, -oc, and -es are for scheduled file transfers in the **fteCreateTransfer** command. If you want to know more information about the scheduling parameters for **fteCreateTransfer**, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/start_new_transfer_cmd.htm

We use two examples to explain the **fteCreateTransfer** command with scheduling:

- Sample 1

  In Example 5-9, the file transferPicture.jpg is transferred from WASHQM's system to NYQM's system. The file transfer is scheduled to take place at 09:00 based on the system time of the source agent's machine and will occur every two hours, four times.

  Example 5-9 Scheduled transfer example 1

  ```
  fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM
  -tb source -ss 09:00 -oi hours -of 2 -oc 4 -df
  /home/wmbadmin/ny_transferPicture.jpg C:\transferPicture.jpg
  ```

  Specify -de parameter: If you do not specify the -de parameter that overwrites the existing file, the file transfers will fail because the second triggered transfer in the same scheduling request happens.

- Sample 2

  In Example 5-10 on page 110, the file transferPicture.jpg is transferred from NYQM's system to MUMBAIQM's system. The file transfer is scheduled to take place at 06:35 based on the system time of the source agent's machine and will occur every five minutes until 07:35.

Example 5-10 Scheduled transfer example 2

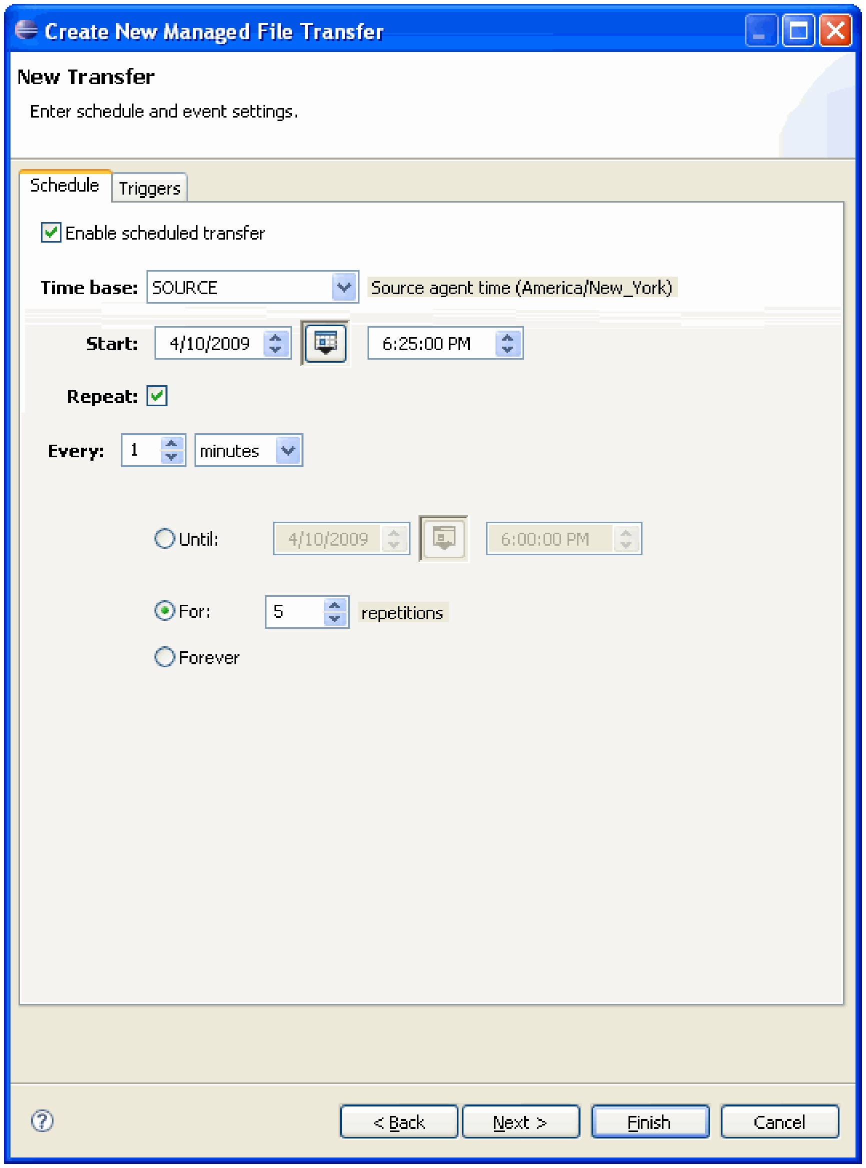
```
fteCreateTransfer -sa NY.AGENT -sm NYQM -da MUMBAI.AGENT -dm
MUMBAIQM -dd /var/mqm -t Binary -cs MD5
-de overwrite -pr 3 -ss 06:35 -tb source -of 5 -oi minutes -es 07:35
-jn MYJOB /home/wmbadmin/ny_transferPicture.jpg
```

Creating a scheduled file transfer with the WebSphere MQ Explorer

In this section, we describe the steps to configure a scheduled file transfer in the WebSphere MQ Explorer:

1. Launch WebSphere MQ Explorer, and connect to the Managed File Transfer wizard.
2. Open a **Create New File Transfer** window.
3. Select and input the initial parameters on the first window, and click **Next** on both the schedule and event setting windows.

We can find the parameters on the window, which include time base, start time, repeat option, repetitive units, repetition times, end time, and so on. In Figure 5-11 on page 111, we set a scheduled transfer that takes place at 2009-04-10 18:25 based on the system time of the source agent's machine, which will occur every one minute, five times.


Create New Managed File Transfer

New Transfer

Enter schedule and event settings.

[✓] Enable scheduled transfer

Time base: SOURCE   Source agent time (America/New_York)

Start: 4/10/2009   6:25:00 PM

Repeat: [✓]

Every: 1 minutes

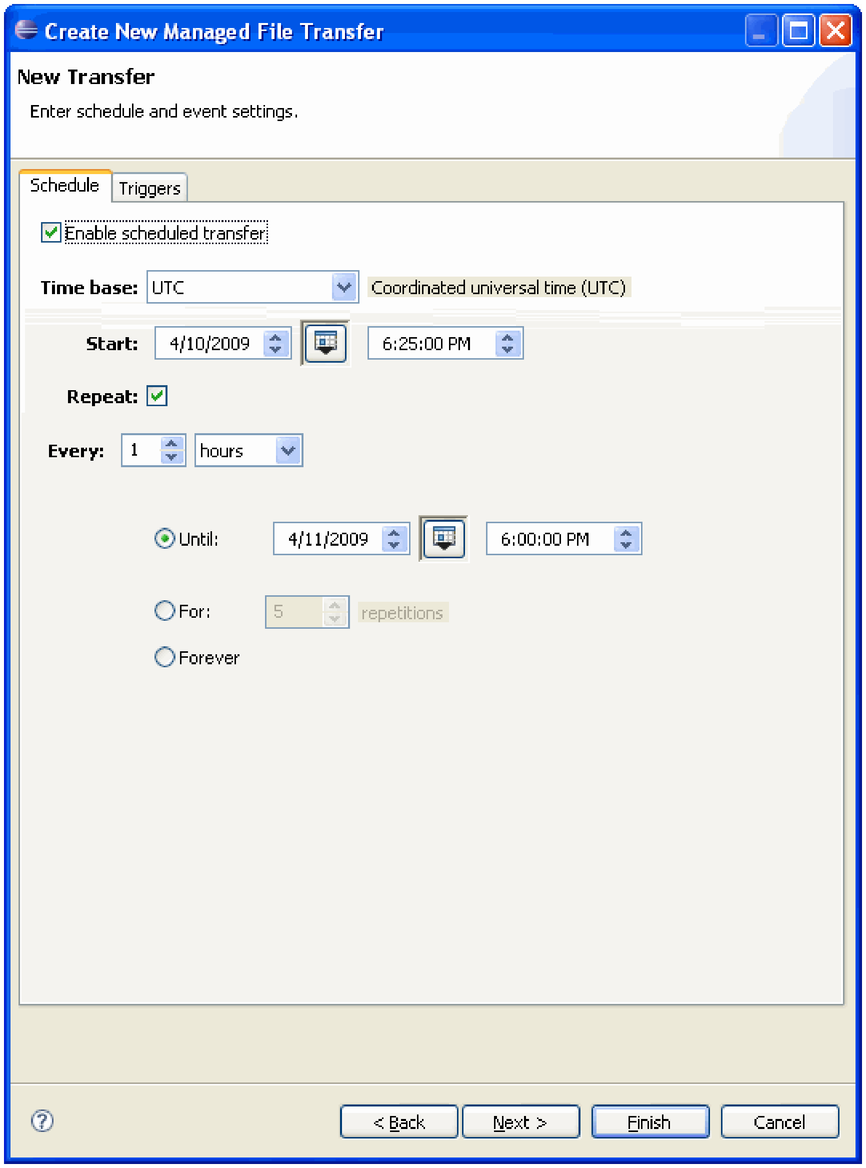
Until: 4/10/2009   6:00:00 PM

[✓ For:] 5 repetitions

Forever

Figure 5-11 Setting a scheduled transfer

In Figure 5-12 on page 112, we configure a scheduled transfer that happens at 2009-04-10 18:25 based on the coordinated universal time, and the transfer will occur every five minutes until 2009-04-11 18:00.


Create New Managed File Transfer

New Transfer

Enter schedule and event settings.

[✓] Enable scheduled transfer

Time base: UTC Coordinated universal time (UTC)

Start: 4/10/2009 6:25:00 PM

Repeat: [✓]

Every: 1 hours

[● Until:] 4/11/2009 6:00:00 PM

For: 5 repetitions

Forever

Figure 5-12 Configuring scheduled transfer that happens at 06:25

Using the file transfer XML request message

In this section, we describe the messages that are involved in scheduled file transfers: the messages to request a file transfer and the messages that are published to the coordination queue manager.

When you submit a scheduled file transfer by placing a message in an agent's command queue, the format of the transfer request message is similar to Example 5-11.

Example 5-11 Scheduled request message example

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>washington.jkhl.com</hostName>
        <userID>wmbadmin</userID>
    </originator>
    <schedule>
       <submit timebase="admin"
            timezone="America/New_York">2008-11-23T22:10</submit>
    </schedule>
    <sourceAgent agent="WASH.AGENT" QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT" QMgr="NYQM"/>
    <transferSet priority="5">
      <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 <schedule> element contents specify which time zone to use (the administrator time zone, the source agent time zone, or coordinated universal time, UTC) and the date and time when the transfer occurs.

If you want to repeat the transfer at a set interval and more than once, specify this information in a <repeat> element in the <schedule> element, for example, the scheduled transfer in Example 5-12 on page 114 causes five transfers to occur, every 10 minutes and starts at 2009-04-11 18:00.

Example 5-12 Example segment of scheduled request message

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
    <originator>
        <hostName>washington.jkhl.com</hostName>
        <userID>wmbadmin</userID>
    </originator>
    <schedule>
      <submit timebase="UTC" timezone="Coordinated Universal
           Time">2009-04-11T18:00</submit>
      <repeat>
         frequency interval="minutes">10</frequency>
         <expireCount>5</expireCount>
      </repeat>
    </schedule>
    <sourceAgent agent="WASH.AGENT" QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT" QMgr="NYQM"/>
    <transferSet priority="5">
      <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>
```

5.2.3 Triggering file transfers

You can set certain trigger conditions on a file transfer that must be true before that transfer can take place. Triggered file transfer is very different than scheduled file transfers. This type of file transfer begins based upon a trigger condition being met. There are three different triggering conditions:

- If a particular file exists on the source agent's system
- If a particular file does not exist on the source agent's system

- If a particular file is over a certain size on the source agent's system (the size can be expressed in bytes, KB, MB, or GB).

Creating a triggered file transfer using the command line

Use the two parameters of -tr and -tl to trigger file transfer in the **fteCreateTransfer** command. If you want to know more information about the file trigger parameters for **fteCreateTransfer**, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/start_new_transfer_cmd.htm

In Example 5-13, the file transferPicture.jpg is transferred from WASH.AGENT to NY.AGENT, on the condition that the file FTELog.txt exists on WASH.AGENT. In Example 5-13, FTE does not print log files because the -tl parameter is set to “no“.

Example 5-13 Triggered file transfer example

```
fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM
-tl no -tr file=exist,C:\IBM\WMQFTE\FTELog.txt -df
/home/wmbadmin/transferPic.jpg C:\transferPicture.jpg
```

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.

In Example 5-14, the file file1.doc is transferred from NY.AGENT to MUMBAI.AGENT, on the condition that:

- Either file A.txt, file B.txt, or both files exist on NY.AGENT
- Either file A.txt, file B.txt, or both files are equal to or larger than 1 GB

Example 5-14 Triggered file transfer example 2

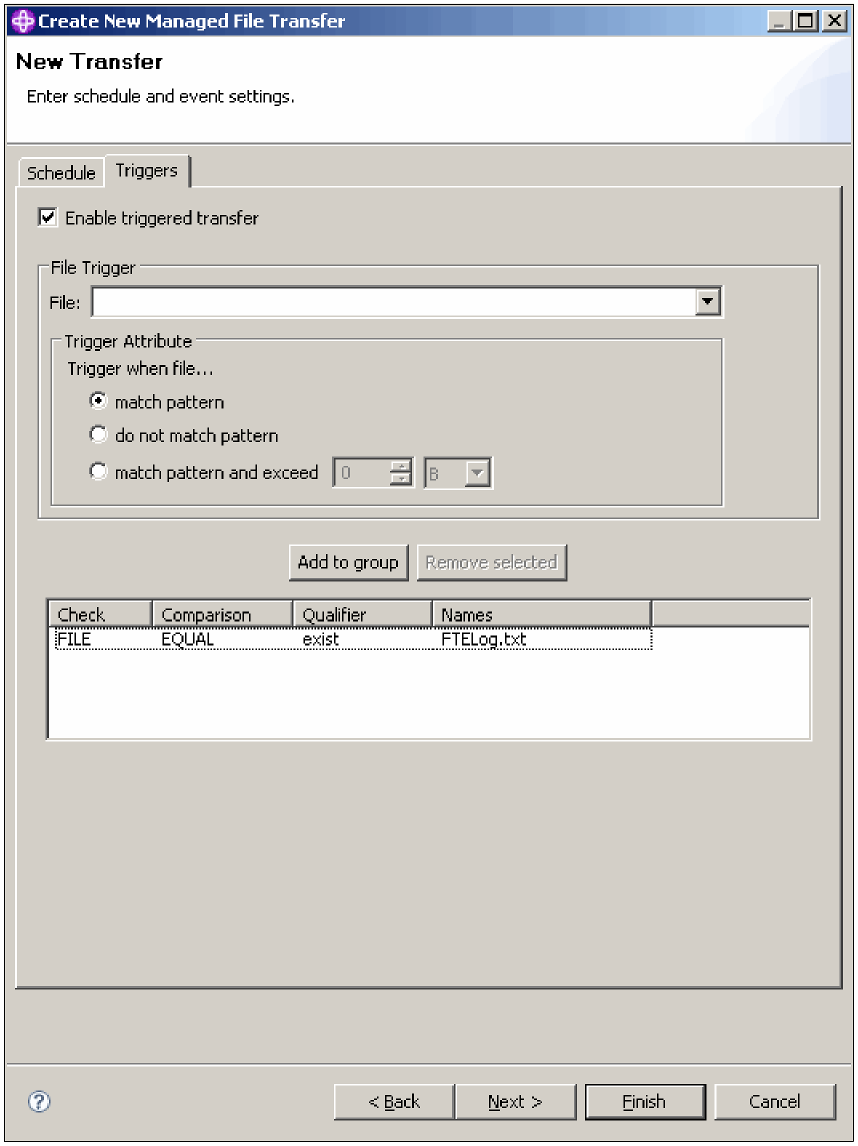
```
fteCreateTransfer -sa NY.AGENT -sm NYQM -da MUMBAI.AGENT -dm MUMBAIQM
-tr file=exist,C:\export\A.txt,C:\export\B.txt -tr
filesize">"=1GB,C:\export\A.txt,C:\export\B.txt -df C:\import\file1.doc
C:\export\file1.doc
```

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 that the scheduling parameters create.

Configuring triggered file transfer in WebSphere MQ Explorer

In the section, we describe the steps to configure triggered file transfer in the WebSphere MQ Explorer. We introduced how to open the Managed File

Transfer wizard and connect to the coordination queue manager. So we can directly switch to the schedule and event setting window, and select the **Triggers** panel. Figure 5-13 shows the trigger conditions and attributes. In Figure 5-13, we also set the triggered file transfer that takes place as FTELog.txt matches in the source agent.



[✓] Enable triggered transfer

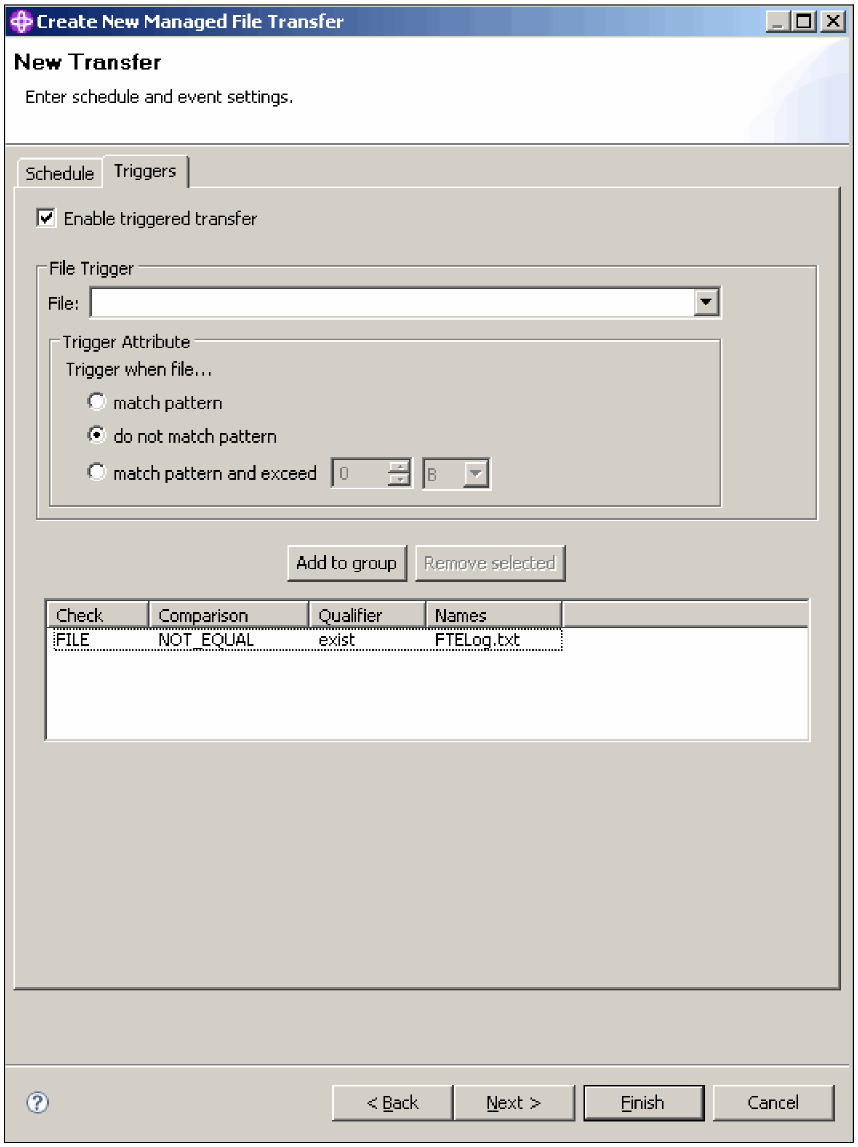
Trigger when file...

[✓ match pattern] do not match pattern / match pattern and exceed 0 B

Check	Comparison	Qualifier	Names
FILE	EQUAL	exist	FTELog.txt

Figure 5-13 Setting trigger conditions and attributes

We change the condition, which happens when FTEError.txt does not match in the source agent, as shown in Figure 5-14.


Create New Managed File Transfer

New Transfer

Enter schedule and event settings.

Schedule | Triggers

[✓] Enable triggered transfer

File Trigger

File:

Trigger Attribute

Trigger when file...

( ) match pattern

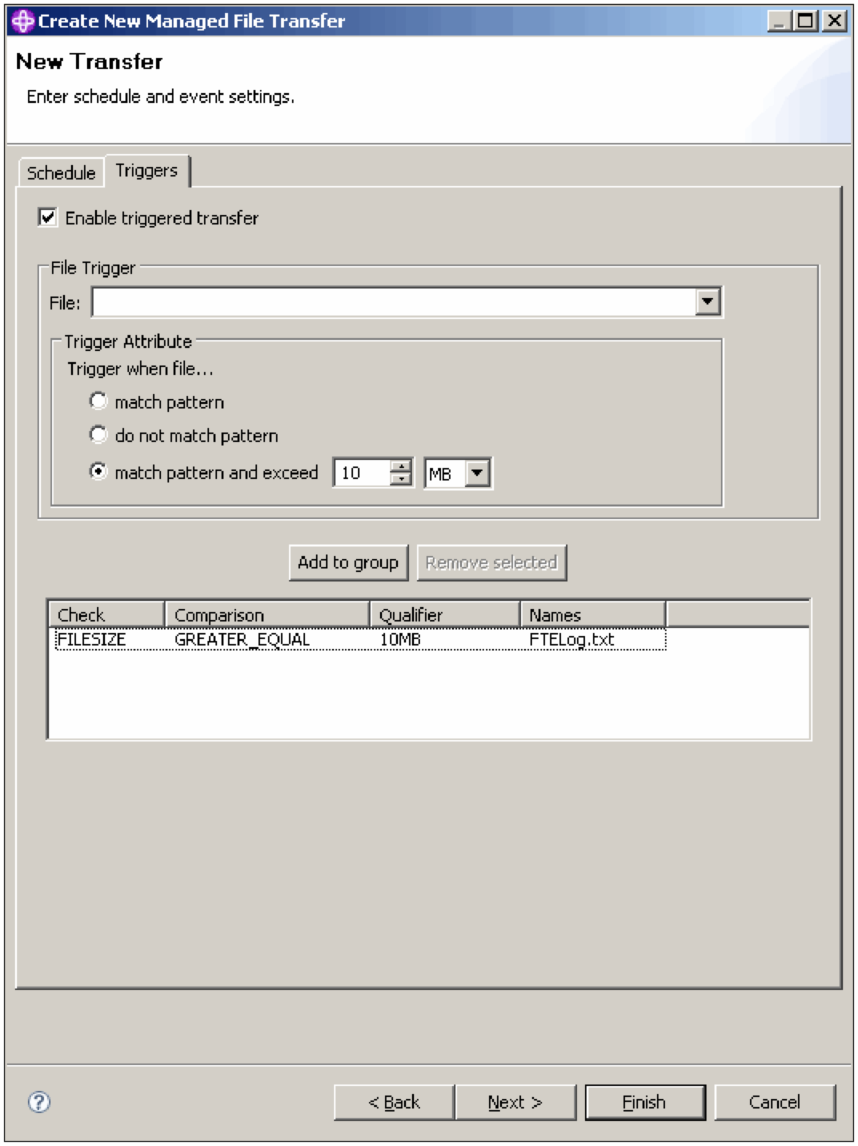
(●) do not match pattern

( ) match pattern and exceed 0 B

Check	Comparison	Qualifier	Names
FILE	NOT_EQUAL	exist	FTELog.txt

Figure 5-14 Triggering transfer when FTEError.txt does no exist in the source agent

Finally, we input the condition of the minimum FTELog.txt size. This optional size unit is MB. When the FTELog.txt file size is more than 10 MB, the triggered transfer occurs in the condition, as shown in Figure 5-15 on page 118.



Create New Managed File Transfer

New Transfer

Enter schedule and event settings.

[✓] Enable triggered transfer

File Trigger

File:

Trigger Attribute

Trigger when file...

- ○ match pattern
- ○ do not match pattern
- ● match pattern and exceed 10 MB

Check	Comparison	Qualifier	Names
FILESIZE	GREATER_EQUAL	10MB	FTELog.txt

Figure 5-15   The trigger occurs when the file size exceeds 10 MB

Using the file transfer XML request message

Example 5-15 shows the part of the XML request message that includes the triggered file transfer condition.

Example 5-15 Example segment of the transfer request message

```
<trigger log="yes">
   <fileExist comparison="="
      value="exist">C:\IBM\WMQFTE\FTELog.txt</fileExist>
</trigger>
```

5.2.4 Creating resource monitors to initialize file transfers

You can monitor a resource, such as a directory, using WebSphere MQ File Transfer Edition so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started. Resource monitors can also start other tasks, such as commands (Ant scripts, executables or JCL).

WebSphere MQ File Transfer Edition has methods to create a resource monitor:

- Command line or shell script
- WebSphere MQ Explorer

Additionally, you can create a resource monitor by placing an XML message on the command queue.

Creating a resource monitor using the command line

You use the **fteCreateMonitor** command to create and then start a resource monitor for a WebSphere MQ File Transfer Edition agent, for example, you can use a resource monitor in the following way: an external application puts a file 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, which copies the files from the known directory to a destination agent. We introduce the command format in Example 5-16.

Example 5-16 The fteCreateMonitor command usage

```
fteCreateMonitor  [-p ConfigurationOptions] [-ma agentName]
                   [-mm queueManager] [-mn monitorName]
                   [-mt task_filename] [-md directory]
                   [-rl recursion_level] [-pi poll_interval]
                   [-pu poll_units] [-tr trigger_condition]
                  [-jn job_name]
```

In the following list, we explain the syntax in Example 5-16 on page 119:

- -ma agentName

  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 queueManager

  Optional. The name of the queue manager that the monitoring agent is connected to. Because the monitoring agent and the source agent must be the same, this queue manager is also your source agent's queue manager.

- -mn monitorName

  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 new monitor with the same name.

- -md directory

  Required. The name of the directory path that you want to monitor.

- -mt task_filename

  Required. The name of the XML document that contains the task definition that you want to carry out when the trigger condition is satisfied.

  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.

- -rl recursion_level

  Optional. The level of monitoring recursion of the root monitoring directory, which is how many levels of subdirectory to navigate down into.

- -pi poll_interval

  Optional. The interval period between each monitor of a directory. If you specify the -pi parameter, you must also specify the -pu parameter.

- -pu poll_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

- -jn job_name

  Optional. Job name reference. A user-defined identifier for the request.

In Example 5-17, the resource monitor JKHLMonitor with a monitoring (and source agent) WASH.AGENT is created with polling every five seconds for file transfers.

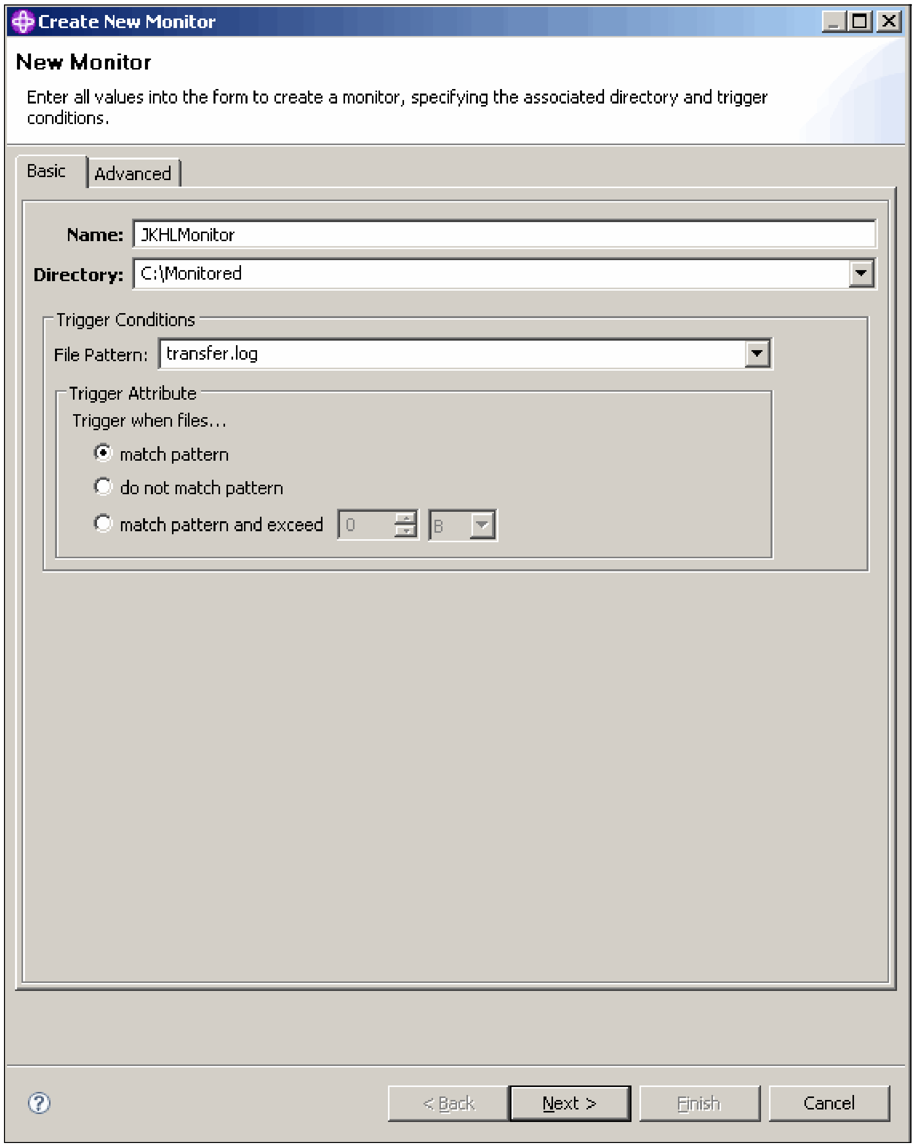
Example 5-17 The fteCreateMonitor command example

```
fteCreateMonitor -ma WASH.AGENT -mm WASHQM -mn JKHLMonitor
      -mt JKHLTransfer.xml -md C:\IBM\WMQFTE\Monitor -pi 5 -pu seconds
      -tr match,trigger.txt
```

Creating a new resource monitor in WebSphere MQ Explorer

To create a new resource monitor in WebSphere MQ Explorer:

1. Launch WebSphere MQ Explorer, and connect to the Managed File Transfer wizard.
2. Start the Create New Monitor window, and select and input the parameters, as shown in Figure 5-16 on page 122.


Create New Monitor

New Monitor

Enter all values into the form to create a monitor, specifying the associated directory and trigger conditions.

Name: JKHLMonitor

Directory: C:\Monitored

Trigger Conditions

File Pattern: transfer.log

Trigger Attribute

Trigger when files...

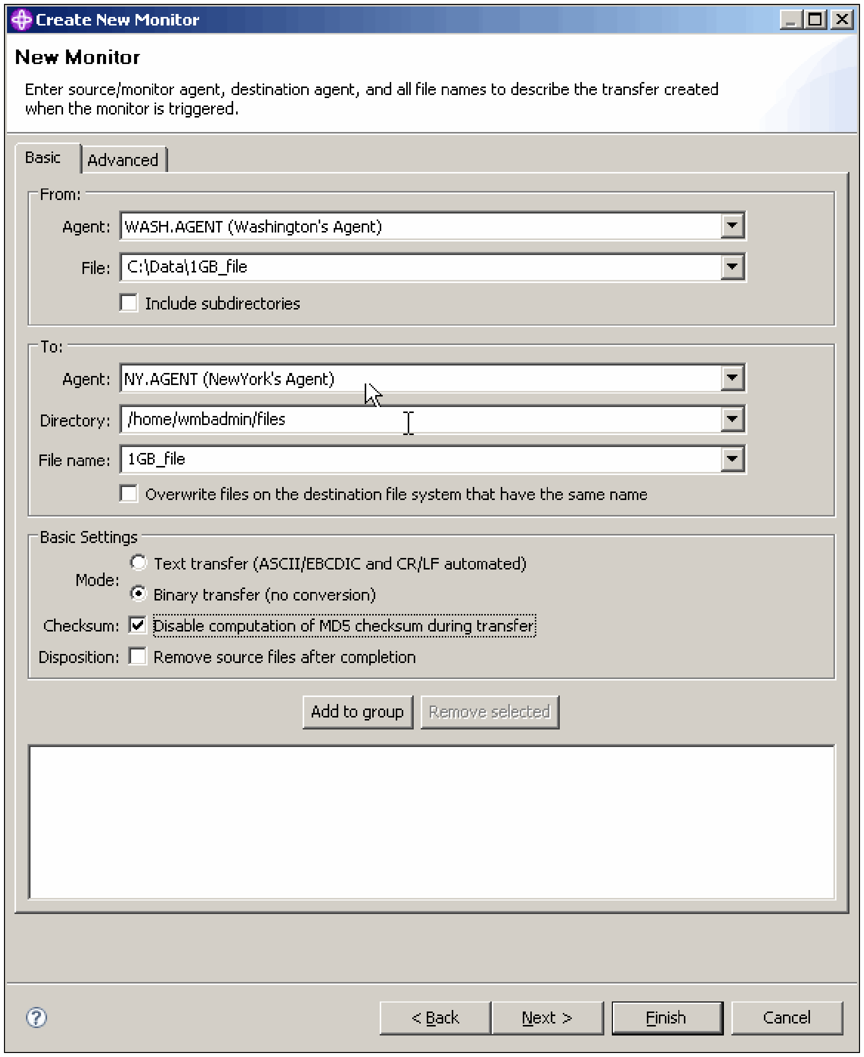
[✓ match pattern]

do not match pattern

match pattern and exceed 0 B

Figure 5-16 Create a resource monitor

3. After you input basic resource information, click **Next**. Figure 5-17 on page 123 shows how to specify the transfer task to be started when the monitor trigger condition is satisfied.


Create New Monitor

New Monitor

Enter source/monitor agent, destination agent, and all file names to describe the transfer created when the monitor is triggered.

From:

Agent: WASH.AGENT (Washington's Agent)

File: C:\Data\1GB_file

[ ] Include subdirectories

To:

Agent: NY.AGENT (NewYork's Agent)

Directory: /home/wmbadmin/files

File name: 1GB_file

[ ] Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ( ) Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ Binary transfer (no conversion)]

Checksum: [✓] Disable computation of MD5 checksum during transfer

Disposition: [ ] Remove source files after completion

Figure 5-17 Setting a triggered file transfer from a resource monitor

4. Click **Finish**. When the transfer.log file is created in the C:\Monitored directory and then detected, a 1 GB file is sent to New York from Washington.

5.3 Administering file transfers

In this section, we describe some administration tasks for file transfers:

- Cancelling file transfers in process
- Listing scheduled file transfers
- Deleting scheduled file transfers
- Deleting resource monitors
- Listing resource monitors

5.3.1 Cancelling file transfers in process

You can use the **fteCancelTransfer** command to cancel a WebSphere MQ File Transfer Edition transfer. Because this is a remote command, you can run the **fteCancelTransfer** command from any system that can connect to the WebSphere MQ network and subsequently route to the agent's queue manager.

If you issue the **fteCancelTransfer** command while that transfer is currently in progress, any files that are already transferred or partially transferred as part of that transfer remain on the destination system and are not deleted. The destination side of the transfer logs that transfer as *cancelled*.

You must set the agent queue manager to be either the source or destination agent for the transfer that you want to cancel and the transfer ID that is from the **fteCreateTransfer** command result. Example 5-18 shows the syntax for the **fteCreateTransfer** command.

Example 5-18 Syntax of the fteCancelTransfer command

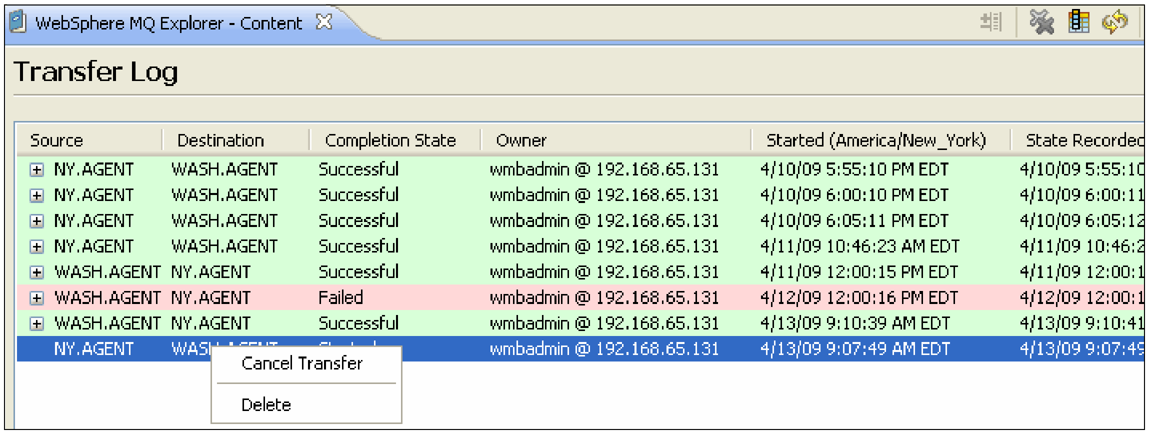
```
fteCancelTransfer [-p ConfigurationOptions]
                  [-m agentQueueManager]
                  -a agentName TransferID
```

In Example 5-19, WASH.AGENT is the source agent for the transfer to be canceled.

Example 5-19 Cancelling the file Transfer

```
fteCancelTransfer -m WASHQM
-a WASH.AGENT 414d512057415348514d202020202020fa9be04920003602
```

You can also cancel the file transfer in process with the WebSphere MQ Explorer, as shown in Figure 5-18 on page 125.



Transfer Log

Source	Destination	Completion State	Owner	Started (America/New_York)	State Recordec
NY.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/10/09 5:55:10 PM EDT	4/10/09 5:55:10
NY.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/10/09 6:00:10 PM EDT	4/10/09 6:00:11
NY.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/10/09 6:05:11 PM EDT	4/10/09 6:05:12
NY.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/11/09 10:46:23 AM EDT	4/11/09 10:46:2
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/11/09 12:00:15 PM EDT	4/11/09 12:00:1
WASH.AGENT	NY.AGENT	Failed	wmbadmin @ 192.168.65.131	4/12/09 12:00:16 PM EDT	4/12/09 12:00:1
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/13/09 9:10:39 AM EDT	4/13/09 9:10:41
NY.AGENT	WAS		wmbadmin @ 192.168.65.131	4/13/09 9:07:49 AM EDT	4/13/09 9:07:49

Figure 5-18 Cancelling the transfer with WebSphere MQ Explorer

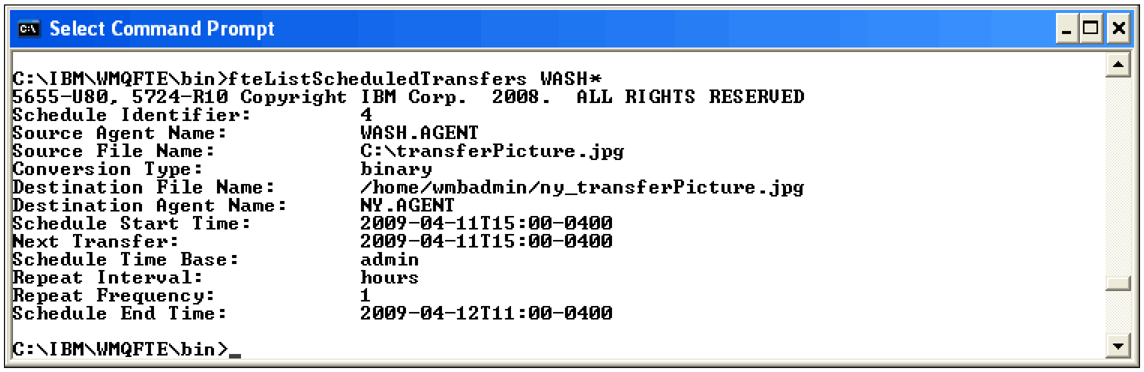
5.3.2 Listing scheduled file transfers

You can use the **fteListScheduledTransfers** command to list all of the WebSphere MQ File Transfer Edition transfers that you previously created, either using the command line or the WebSphere MQ Explorer. You can list all scheduled transfers based on source agent names or based on the coordination queue manager. Example 5-20 shows the syntax for the **fteListScheduledTransfers** command.

Example 5-20 The fteListScheduledTransfers Syntax

```
fteListScheduledTransfers [-p ConfigurationOptions] Pattern
```

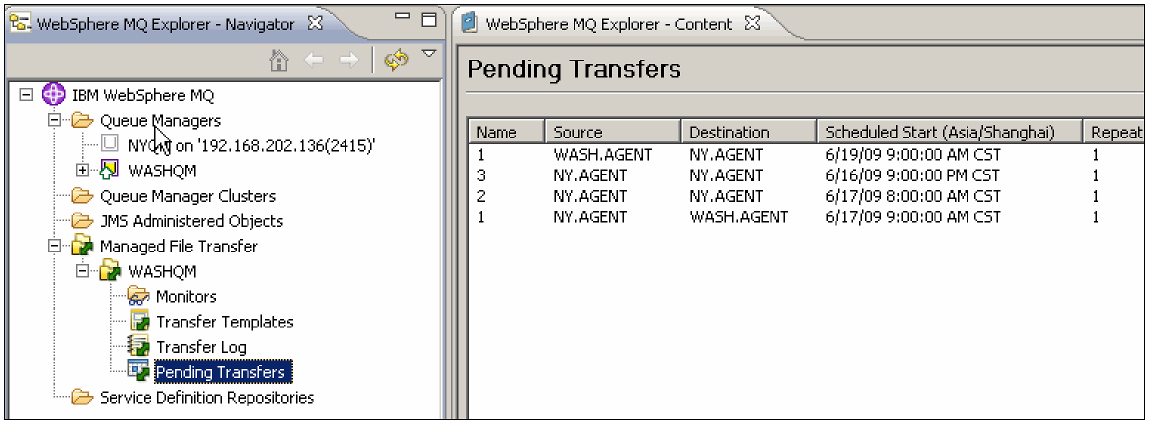
In Example 5-20, the scheduled transfers with source agents that match the pattern WASH* are listed. The example command in Example 5-20 produces the output shown in Figure 5-19 on page 126. The schedule start time and next transfer time are displayed in local time (America/New_York).



```
C:\IBM\WMQFTE\bin>fteListScheduledTransfers WASH*
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
Schedule Identifier:          4
Source Agent Name:            WASH.AGENT
Source File Name:             C:\transferPicture.jpg
Conversion Type:              binary
Destination File Name:        /home/wmbadmin/ny_transferPicture.jpg
Destination Agent Name:       NY.AGENT
Schedule Start Time:          2009-04-11T15:00-0400
Next Transfer:                2009-04-11T15:00-0400
Schedule Time Base:           admin
Repeat Interval:              hours
Repeat Frequency:             1
Schedule End Time:            2009-04-12T11:00-0400

C:\IBM\WMQFTE\bin>_
```

Figure 5-19 The issued output of the fteListScheduledTransfers command

You can list the scheduled transfers in WebSphere MQ Explorer, as shown in Figure 5-20.



Pending Transfers

Name	Source	Destination	Scheduled Start (Asia/Shanghai)	Repeat
1	WASH.AGENT	NY.AGENT	6/19/09 9:00:00 AM CST	1
3	NY.AGENT	NY.AGENT	6/16/09 9:00:00 PM CST	1
2	NY.AGENT	NY.AGENT	6/17/09 8:00:00 AM CST	1
1	NY.AGENT	WASH.AGENT	6/17/09 9:00:00 AM CST	1

Figure 5-20 The scheduled transfer list on WebSphere MQ Explorer

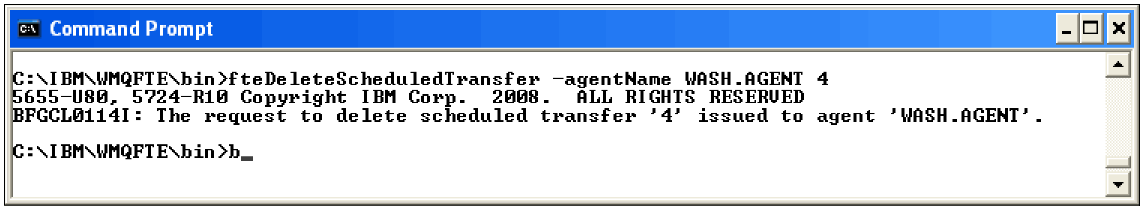
5.3.3 Deleting scheduled file transfers

You can use the **`fteDeleteScheduledTransfer`** command to delete a WebSphere MQ File Transfer Edition scheduled transfer that you previously created. You can use either the command line or the WebSphere MQ Explorer. Example 5-21 on page 127 shows the syntax for the **`fteDeleteScheduledTransfer`** command.

Example 5-21 Syntax of the fteDeleteScheduledTransfer command

```
fteDeleteScheduledTransfer [-p ConfigurationOptions]
                           [-m agentQueueManager]
                           -agentName agentName ScheduleID
```

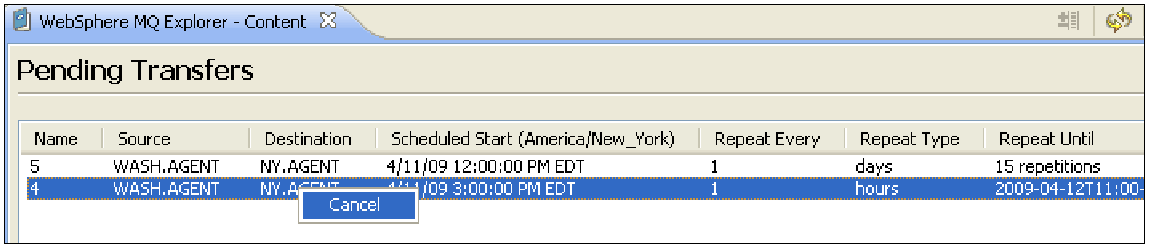
In Example 5-21, a scheduled transfer on source agent WASH.AGENT with the ID 4 is deleted, as shown in Figure 5-21.



```
C:\IBM\WMQFTE\bin>fteDeleteScheduledTransfer -agentName WASH.AGENT 4
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0114I: The request to delete scheduled transfer '4' issued to agent 'WASH.AGENT'.

C:\IBM\WMQFTE\bin>b_
```

Figure 5-21 The output of the fteDeleteScheduledTransfer command

In the WebSphere MQ Explorer, you can select one scheduled transfer item, right-click the item, and click **Cancel**, as shown in Figure 5-22. The selected item disappears after it is cancelled.



Name	Source	Destination	Scheduled Start (America/New_York)	Repeat Every	Repeat Type	Repeat Until
5	WASH.AGENT	NY.AGENT	4/11/09 12:00:00 PM EDT	1	days	15 repetitions
4	WASH.AGENT	NY.A	/09 3:00:00 PM EDT	1	hours	2009-04-12T11:00-

Figure 5-22 Deleting the scheduled transfers

5.3.4 Deleting resource monitors

You can use the **`fteDeleteMonitor`** command to delete an existing WebSphere MQ File Transfer Edition resource monitor after the monitor is stopped. You can issue this command against either the source or destination agent for the transfer. Example 5-22 shows the syntax for the **`fteDeleteMonitor`** command.

Example 5-22 The syntax of the fteDeleteMonitor command

```
fteDeleteMonitor   [-p ConfigurationOptions]
                   [-ma agentName]
                   [-mm queueManager]
                   [-mn monitorName]
```

In Example 5-23, the resource monitor JKHLMonitor with a monitoring (and source) agent WASH.AGENT is deleted.

Example 5-23 Deleting a resource monitor

```
fteDeleteMonitor -ma WASH.AGENT -mm WASHQM -mn JKHLMonitor
```

5.3.5 Listing resource monitors

The **fteListMonitors** command lists existing resource monitors, as shown in Example 5-24. You can filter the command output by specifying an agent name and a resource monitor name.You can run the **fteListMonitors** command from any system that can connect to the WebSphere MQ network and subsequently route to the monitoring agent's queue manager.

Example 5-24 Displaying the resource monitor list

```
fteListMonitors [-p ConfigurationOptions]
                [-ma agentName]
                [-mn monitorName]
                [-v]
```

The following list explains the syntax for the **fteListMonitors** command:

- -p ConfigurationOptions

  Optional. Determines the set of configuration options that are used to define the coordination queue manager for this call. 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.

- -ma agentName

  Optional. Filters the resource monitor with the given agent name. The default is to list all resource monitors for all agents that are associated with the coordination queue manager.

- -mn monitorName

  Optional. Filters for the named resource monitor. The default is to list resource monitors for all agents that are associated with the coordination queue manager.

- -v

  Optional. The verbose mode shows the status of the monitor, the monitored resource, the match type, and the trigger condition.

Example 5-25 lists all of the resource monitors that are associated with the monitoring agent WASH.AGENT.

Example 5-25 List the resource monitor on WASH.AGENT

```
fteListMonitors -ma WASH.AGENT -v
```

5.4 Administering WebSphere MQ File Transfer Edition agents

An FTE agent is the main transfer component running on a WebSphere MQ queue manager. Agents connect to queue managers by binding mode or client mode. If you create or manage the agents, you must first set the agent's coordination queue manager and command queue manager. In this section, we discuss how to manage the FTE agents.

5.4.1 Setting up the coordination queue manager

Use the **fteSetupCoordination** command to create properties files and the coordination queue manager directory for WebSphere MQ File Transfer Edition. We can issue the **fteSetupCoordination** command to create the following WebSphere MQ File Transfer Edition objects:

- Coordination queue manager directory
- Data directory (if this does not already exist)
- wmqfte.properties file
- coordination.properties file

Example 5-26 shows the syntax for the **fteSetupCoordination** command.

Example 5-26 The fteSetupCoordination command syntax

```
fteSetupCoordination -coordinationQMgr qmgr_name
                      [-coordinationQMgrHost qmgr_host]
                      [-coordinationQMgrPort qmgr_port]
                      [-coordinationQMgrChannel qmgr_channel]
                      [-f] [-default]
```

To get more information about the parameters for the **fteSetupCoordination**, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/setup_coord_cmd.htm

The **fteSetupCoordination** command also provides you with the MQSC commands that are listed in Example 5-27. You must run the commands against your coordination queue manager to configure WebSphere MQ File Transfer Edition. The MQSC commands create a queue, a topic, a topic string, update a namelist, and set the coordination queue manager's PSMODE attribute to ENABLED.

Example 5-27 MQSC command on the coordination queue manager

```
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)
```

In Example 5-28, the required objects are set up for a coordination queue manager called WASH.QM, which is connected to it in client mode.

Example 5-28 The fteSetupCoordination command example

```
fteSetupCoordination -coordinationQMgr WASHQM
   -coordinationQMgrHost raleigh.ibm.com -coordinationQMgrPort 1415
   -coordinationQMgrChannel SYSTEM.DEF.SVRCONN
```

5.4.2 Setting up the command queue manager

The **fteSetupCommands** command, shown in Example 5-29 on page 131, creates the command.properties file. This properties file specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands.

You can use the **fteSetupCommands** command to create a command.properties file in the coordination queue manager directory. The command uses the install.properties and wmqfte.properties files to determine where to locate the command.properties file.

Coordination queue manager: Ensure that you already created and configured a coordination queue manager before you issue the **fteSetupCommands** command.

Example 5-29 The fteSetupCommands command syntax

```
fteSetupCommands -connectionQMgr qmgr_name
               [-connectionQMgrHost qmgr_host]
               [-connectionQMgrPort qmgr_port]
               [-connectionQMgrChannel qmgr_channel]
               [-p ConfigurationOptions] [-f]
```

To get more information about the parameters for the **fteSetupCommands** command, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/setup_cmds_cmd.htm

Example 5-30 shows an example of using the **fteSetupCommands** command.

Example 5-30 The fteSetupCommands command example

```
fteSetupCommands -connectionQMgr WASHQM -connectionQMgrHost
   raleigh.ibm.com -connectionQMgrPort 1415 -connectionQMgrChannel
   SYSTEM.DEF.SVRCONN
```

5.4.3 Changing the default configuration

Your default WebSphere MQ File Transfer Edition configuration options are established during installation and are based on your default coordination queue manager. Using the **fteChangeDefaultConfigurationOptions** command, shown in Example 5-31, you can change the default configuration options that are defined in the wmqfte.properties file. If you change these configuration options, commands are run against the coordination queue manager that you used as input for configuration_options by default. You can override this default by specifying a different queue manager using the **fteSetupCommands** command.

Example 5-31 The fteChangeDefaultConfigurationOptions command syntax

```
fteChangeDefaultConfigurationOptions ConfigurationOptions
```

In Example 5-32 on page 132, the default configuration options are changed to NYQM.

Example 5-32 The command example

```
fteChangeDefaultConfigurationOptions NYQM
```

5.4.4 Creating an agent

The **fteCreateAgent** command creates an agent and its associated configuration. Example 5-33 shows the command syntax for **fteCreateAgent**.

Example 5-33 The syntax of the fteCreateAgent command

```
fteCreateAgent -agentName agent_name -agentQMgr agent_qmgr
              [-agentQMgrHost agent_qmgr_host]
              [-agentQMgrPort agent_qmgr_port]
              [-agentQMgrChannel agent_qmgr_channel]
              [-agentDesc agent_description]
              [-p ConfigurationOptions] [-f]
```

To get more information about all of the parameters for the **fteCreateAgent** command, refer to:

`http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/create_agent_cmd.htm`

This command provides you with the MQSC commands that you must run against your agent's queue manager to create the following agent queues:

- 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: <configuration_directory>\<coordination_qmgr>\agents\<agent_name>\<agent_name>_create.mqsc

In Example 5-34 on page 133, MUMBAI.AGENT is created with an agent queue manager MUMBAIQM and uses the default coordination queue manager.

Example 5-34 The fteCreateAgent command example on client mode

```
fteCreateAgent -agentName MUMBAI.AGENT -agentQMgr MUMBAIQM
   -agentQMgrHost 192.168.201.100 -agentQMgrPort 3415
   -agentQMgrChannel MUMBAI.CHANNEL
```

5.4.5 Deleting an agent

The **fteDeleteAgent** command, Example 5-35, deletes a WebSphere MQ File Transfer Edition agent and its configuration. You must stop the agent with the **fteStopAgent** command before you run the **fteDeleteAgent** command.

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 five system queues, which are:

- 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>

Example 5-35 The syntax of the fteDeleteAgent command

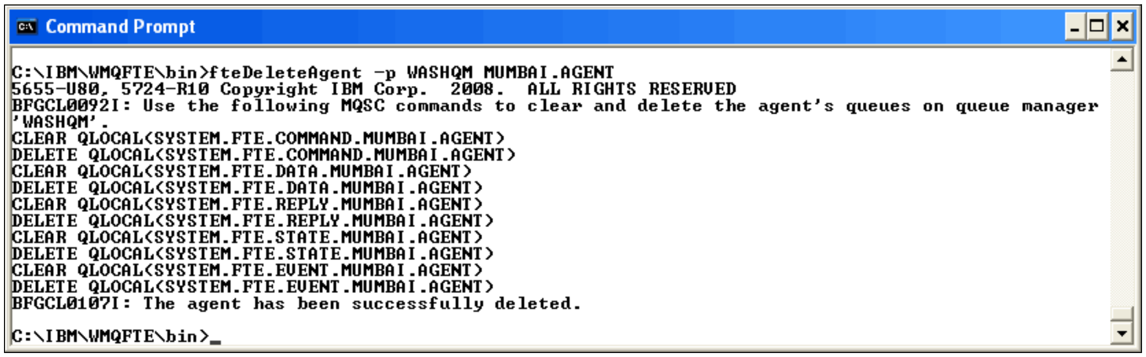
```
fteDeleteAgent -p ConfigurationOptions agentName
```

In Example 5-36, MUMBAI.AGENT and its configuration on coordination queue manager WASHQM are deleted.

Example 5-36 The fteDeleteAgent command

```
fteDeleteAgent -p WASHQM MUMBAI.AGENT
```

The command in Example 5-36 outputs the MQSC commands to delete the agent's three queues, as seen in Figure 5-23 on page 134.



```
C:\IBM\WMQFTE\bin>fteDeleteAgent -p WASHQM MUMBAI.AGENT
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0092I: Use the following MQSC commands to clear and delete the agent's queues on queue manager
'WASHQM'.
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.MUMBAI.AGENT)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.MUMBAI.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.DATA.MUMBAI.AGENT)
DELETE QLOCAL(SYSTEM.FTE.DATA.MUMBAI.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.MUMBAI.AGENT)
DELETE QLOCAL(SYSTEM.FTE.REPLY.MUMBAI.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.STATE.MUMBAI.AGENT)
DELETE QLOCAL(SYSTEM.FTE.STATE.MUMBAI.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.MUMBAI.AGENT)
DELETE QLOCAL(SYSTEM.FTE.EVENT.MUMBAI.AGENT)
BFGCL0107I: The agent has been successfully deleted.

C:\IBM\WMQFTE\bin>_
```

Figure 5-23 The output of the fteDeleteAgent command with -p option

5.4.6 Listing agents

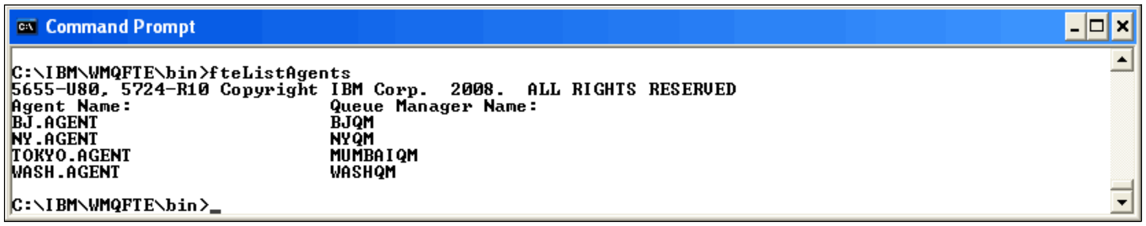
You can list the agents that are registered with a particular queue manager using the command line or the WebSphere MQ Explorer. You can use the **fteListAgents** command, shown in Example 5-37, to list all of the WebSphere MQ File Transfer Edition agents that are registered with a particular coordination queue manager from the command line. The following details for each agent are directed to the standard output device:

- Agent name
- Agent queue manager

Example 5-37 The syntax of the fteListAgents command

```
fteListAgents [-p ConfigurationOptions] [Pattern]
```

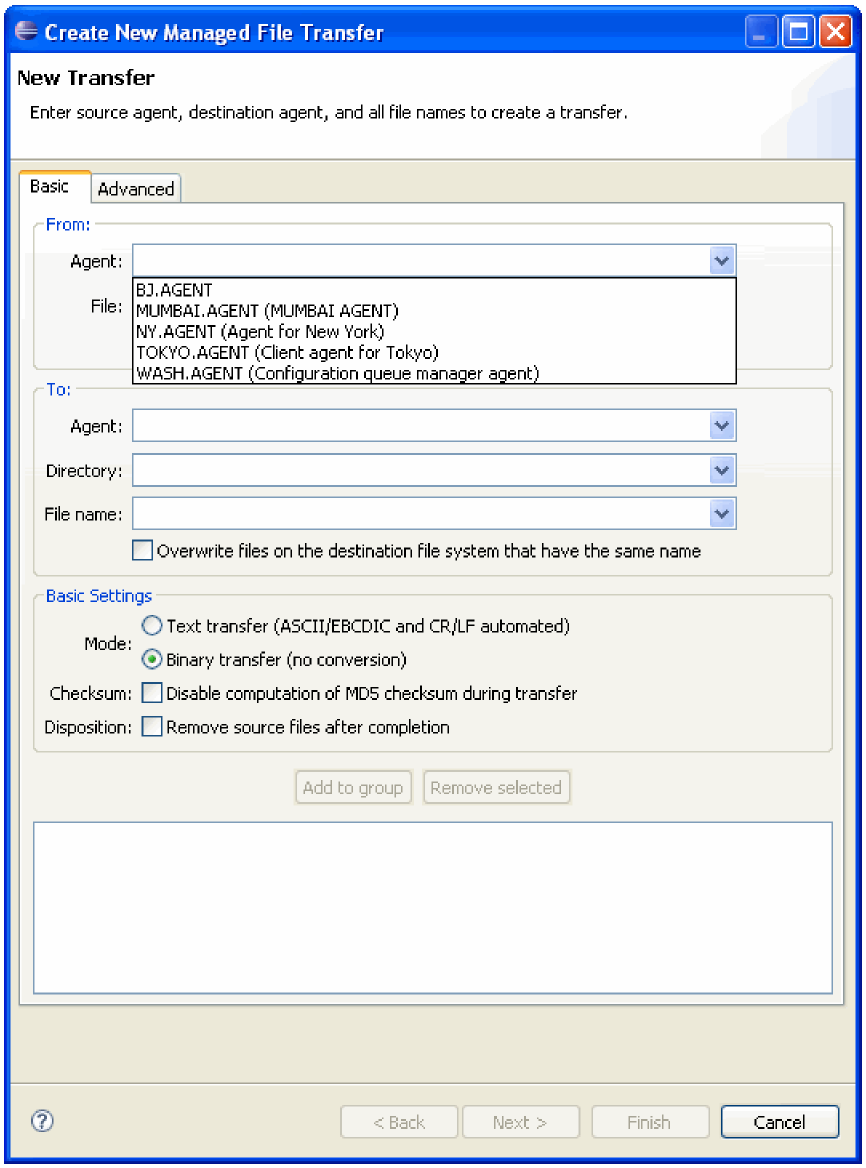
In Example 5-37, all of the agents that are registered on the queue manager are detailed in the configuration options, as shown in Figure 5-24.



```
C:\IBM\WMQFTE\bin>fteListAgents
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
Agent Name:                  Queue Manager Name:
BJ.AGENT                     BJQM
NY.AGENT                     NYQM
TOKYO.AGENT                  MUMBAIQM
WASH.AGENT                   WASHQM

C:\IBM\WMQFTE\bin>_
```

Figure 5-24 The example of the fteListAgents command

You can also view the agent list in the WebSphere MQ Explorer. To list agents using the WebSphere MQ Explorer. To start the Create New Managed Transfer wizard, right-click an individual coordination queue manager, and then click **New Transfer**. The Agent fields in the **To:** and **From:** sections of the Basic tab list the agents that are registered with an individual coordination queue manager. See Figure 5-25.


Create New Managed File Transfer

New Transfer

Enter source agent, destination agent, and all file names to create a transfer.

Basic | Advanced

From:

Agent:

File:

BJ.AGENT
MUMBAI.AGENT (MUMBAI AGENT)
NY.AGENT (Agent for New York)
TOKYO.AGENT (Client agent for Tokyo)
WASH.AGENT (Configuration queue manager agent)

To:

Agent:

Directory:

File name:

☐ Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ○ Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ Binary transfer (no conversion)]

Checksum: ☐ Disable computation of MD5 checksum during transfer

Disposition: ☐ Remove source files after completion

Figure 5-25 Listing the agents

5.4.7 Displaying agent details

You can execute the **`fteShowAgentDetails`** command to display the details of a particular FTE agent. These are the details that are stored by its WebSphere MQ File Transfer Edition coordination queue manager. The agent details are directed to the standard output device. The **`fteShowAgentDetails`** command lists the following details for a particular WebSphere MQ File Transfer Edition agent:

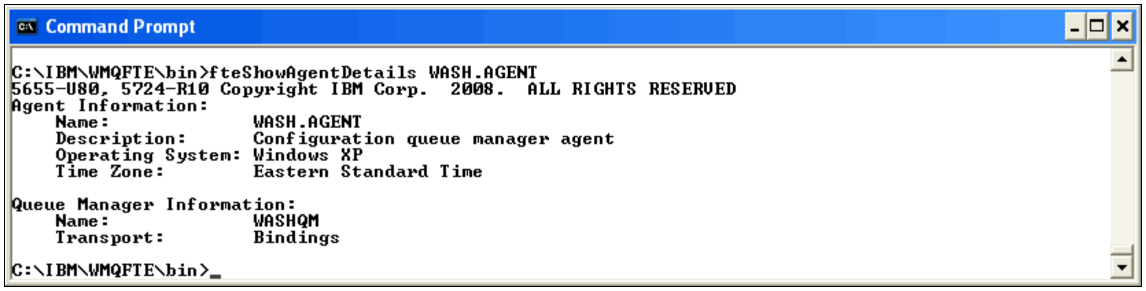
- Agent name
- Agent description
- Operating system
- Agent time zone
- Agent queue manager
- Agent's queue manager transport type
- Agent's queue manager host name
- Agent's queue manager port (applies to client transport mode only)
- Agent's queue manager channel (applies to client transport mode only)

Example 5-38 shows the syntax for the **`fteShowAgentDetails`** command.

Example 5-38 The syntax of the fteShowAgentDetails command

```
fteShowAgentDetails [-p ConfigurationOptions] AgentName
```

In **`fteShowAgentDetails`** command, the details for agent WASH.AGENT are displayed, as shown in Figure 5-26.



```
C:\IBM\WMQFTE\bin>fteShowAgentDetails WASH.AGENT
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
Agent Information:
    Name:             WASH.AGENT
    Description:      Configuration queue manager agent
    Operating System: Windows XP
    Time Zone:        Eastern Standard Time

Queue Manager Information:
    Name:             WASHQM
    Transport:        Bindings

C:\IBM\WMQFTE\bin>_
```

Figure 5-26 The output of the fteShowAgentDetails command

5.4.8 Starting an agent

You must start an agent before you can use it to perform file transfers. Use the **`fteStartAgent`** command to start a WebSphere MQ File Transfer Edition agent. The **`fteStartAgent`** command starts an agent on the system where you issue the command. Because this command is issued locally, you cannot start an agent on a remote system.

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. Example 5-39 shows the syntax for the **fteStartAgent** command.

Example 5-39 The fteStartAgent command syntax

```
fteStartAgent [-F] [-p ConfigurationOptions] AgentName
```

To get more information about all of the parameters for the **fteStartAgent** command, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/start_agent_cmd.htm

In Example 5-40 (for Windows), WASH.AGENT is started and runs in the foreground.

Example 5-40 Starting WASH.AGENT in foreground

```
fteStartAgent -F WASH.AGENT
```

In Example 5-41 (for UNIX and Linux systems), NY.AGENT is started using a non-default coordination queue manager, WASHQM.

Example 5-41 Starting NY.AGENT with non-default coordination QM

```
./fteStartAgent -p WASHQM NY.AGENT
```

5.4.9 Stopping an agent

You can stop an agent using the **fteStopAgent** command. 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 stops, you cannot use that agent to transfer files until you restart it. Stopping an agent also stops any resource monitors that are associated with that agent.

This command is issued remotely, so you can run the **fteStopAgent** command from any system that can connect to the WebSphere MQ network and subsequently route to the agent's queue manager.

Example 5-42 on page 138 shows the syntax for the **fteStopAgent** command.

Example 5-42 The syntax of the fteStopAgent command

```
fteStopAgent [-p ConfigurationOptions] [-m QueueManager] [-i] AgentName
```

To get more information about all of the parameters for the **fteStopAgent** command, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/stop_agent_cmd.htm

In Example 5-43, the agent NY.AGENT on queue manager NYQM is stopped. The -m parameter is used because this queue manager that NY.AGENT is connected to differs from the queue manager that is specified by the set of configuration options.

Example 5-43 The fteStopAgent command example

```
fteStopAgent -m NYQM NY.AGENT
```

5.4.10 Cleaning an agent

If you want to delete all of the messages from the agent's persistent and non-persistent queues, you can use the **fteCleanAgent** command to clean up a WebSphere MQ File Transfer Edition agent's queues. This command does not start the agent. Run this command on an agent that is stopped. If you try to run **fteCleanAgent** on an agent that is currently running, you will receive an error. 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. So you must stop the agent before you issue this command. Example 5-44 shows the syntax for the **fteCleanAgent** command.

Example 5-44 The syntax of the fteCleanAgent command

```
fteCleanAgent [-p ConfigurationOptions] AgentName
```

In Example 5-45 WASH.AGENT queues are cleaned up.

Example 5-45 Cleaning up the agent

```
fteCleanAgent WASH.AGENT
```

If you are running the **fteCleanAgent** command on an agent that is connected to its queue manager in bindings mode, and the agent 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 will remove this handle, and you can reissue **fteCleanAgent**.

5.4.11 Setting agent trace level

You can use this command to switch agent trace on and off or 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: <configuration_directory>\<coordination_qmgr_name>\agents\<agent_name>\logs

Because running trace can impact your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Example 5-46 shows the syntax for the **fteSetAgentTraceLevel** command.

Example 5-46 The syntax of the fteSetAgentTraceLevel command

```
fteSetAgentTraceLevel [-p ConfigurationOptions] -traceLevel traceLevel
                     [-traceClasses traceClasses]
                     [-agentQMgr agentQMgr] agentName
```

To get more information about all of the parameters for the **fteSetAgentTraceLevel** command, refer to:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/agent_trace_cmd.htm

In Example 5-47, the trace level is set to all for the classes com.ibm.wmqfte.agent.Agent and com.ibm.wmqfte.cmdhandler for WASH.AGENT.

Example 5-47 The example of setting agent trace level

```
fteSetAgentTraceLevel -traceLevel moderate
   -traceClasses com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler
   WASH.AGENT
```

5.5 Extended management in WebSphere MQ File Transfer Edition

WebSphere MQ File Transfer Edition supports extended functions. You can use Apache Ant through the FTE-provided **`fteAnt`** command. This command can run many tasks, including filecopy, filemove, filedelete, and multi-steps transfer. If you try to test the connection with agents, you can use the **`ftePingAgent`** command. You can also issue the **`fteCreateMonitor`** command for triggering file transfer.

5.5.1 Using the Ant command for file transfers

You can integrate the file transfer function into the Apache Ant tool in WebSphere MQ File Transfer Edition. By using the file transfer Ant tasks from your Ant scripts, you can coordinate complex file transfer operations from an interpreted scripting language.

WebSphere MQ File Transfer Edition provides a number of Ant tasks that you can use to access file transfer capabilities, such as the tasks in the following list:

- awaitoutcome
- call
- cancel
- filecopy
- filemove
- ignoreoutcome
- ping
- uuid

The following nested parameters describe nested sets of elements, which are common across several of the supplied Ant tasks:

- filespec
- metadata
- program invocation

Note: The **`fteAnt`** command attributes are as described in the WebSphere MQ File Transfer Edition Information Center under the Ant section.

Example 5-48 on page 141 describes the **`fteAnt`** command syntax.

Example 5-48 The syntax of the fteAnt command

```
fteAnt [options] [target [target2 [target3] ...]]
```

In Example 5-49, the file copy action in the Ant script fte_copy.xml is run and the command writes debugging output to standard out.

Example 5-49 The fteAnt command example

```
   fteAnt -d -f fte_copy.xml
```

The fte_copy.xml information is listed in Example 5-50.

Example 5-50 filecopy action description as XML format

```
<?xml version="1.0" ?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs">
   <fte:filecopy cmdqm="WASHQM@washington@1415@SYSTEM.DEF.SVRCONN"
               src="WASH.AGENT@WASHQM" dst="NY.AGENT@NYQM"
               rcproperty="copy.result origuser=”wmbadmin”">
   <fte:filespec srcfilespec="C:\WMQFTE.jpg"
               dstfile="/home/wmbadmin/WMQFTE.jpg" />
  </fte:filecopy>
</project>
```

5.5.2 Testing connectivity with the FTE ping command

The **ftePingAgent** command sends a no-operation transfer to an agent to determine if the agent is active and able to process. Example 5-51 shows the syntax for the **ftePingAgent** command.

Example 5-51 The syntax of the ftePingAgent command

```
ftePingAgent [-p ConfigurationOptions] [-m QueueManager] [-w] agentName
```

In the following list, we explain the syntax in Example 5-51.

- -w

  Optional. Specifies whether to wait for the agent to respond. If you specify the -w parameter, the command waits for the request to complete before the command stops. The default is to wait up to five seconds for the agent to respond.

- AgentName

  Required. The name of the WebSphere MQ File Transfer Edition agent to ping.

Example 5-52 sends a ping request to NY.AGENT hosted on NY.QM. The ping request waits for the agent to respond.

Example 5-52 The ftePingAgent command usage

```
ftePingAgent -m NYQM -w NY.AGENT
```

5.5.3 Configuring resource monitor tasks to invoke commands

Resource monitors are not limited to performing file transfers. You can also configure the monitor to invoke other commands from the monitor agent, which includes executable programs, Ant scripts, or JCL. To do this, the monitor task definition XML needs to be edited to include one or more command elements with corresponding command call parameters, such as arguments and properties. Example 5-53 shows a task definition XML document.

Example 5-53 Task definition XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>washington.jkhl.com</hostName>
      <userID>wmbadmin</userID>
      <mqmdUserID>wmbadmin</mqmdUserID>
    </originator>
    <agent QMgr="WASHQM" agent="WASH.AGENT"/>
    <reply QMGR="WASHQM">reply</reply>
    <transferSet priority="1">
      <metaDataSet>
        <metaData key="accountID">1234567890</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}"/>
```

```
          <property name="trigger.path" value="${FilePath}"/>
        </command>
      </call>
    </transferSet>
    <job>
      <name>JKHL_JOBCLEAN</name>
    </job>
  </managedCall>
</request>
```

The task definition must start with a <request> root element. The next child element must then be either <managedTransfer> or <managedCall>. You typically choose <managedCall> when there is a single command or script to execute, and you choose <managedTransfer> if you want the task to include a file transfer and optionally up to four command calls.

The <agent> element specifies the WebSphere MQ File Transfer Edition agent that is configured with the named Ant script on its commandPath.

The <command><call> structure defines the executable or script that you want to run. The command takes an optional *type* attribute that can have one of the values in Table 5-3.

Table 5-3 The type attribute value

Value	Description
antscript	Runs an Ant script in a separate JVM™
executable	Invokes an executable program
jcl	Invokes a piece of JCL

If the <type> attribute is omitted, the default value <executable> is used.

The <name> attribute specifies the name of the Ant script, executable, or JCL that you want to invoke, without any path information. The agent searches for the script or program in the locations that the commandPath property specifies in the agent's agent.properties file.

For an Ant script, you typically specify <target> and <property> elements. The target element values must match the target names in the Ant script.

The administrator defines and starts the resource monitor as normal using the task definition XML document that includes the <managedCall> element, as the Example 5-54 shows.

Example 5-54 Invoking a command in a resource monitor

```
fteCreateMonitor -ma WASH.AGENT -mm WASHQM -md C:\Monitored -mn
      JKHLMonitor -mt C:\cleanuptask.xml -pi 30 -pu seconds
      -tr match,transfer.log
```

Part 2

Usage scenarios

6

The business scenario that we use in this book

In this chapter, we introduce JKHL Travel (JKHL), the fictional customer in our simplified business scenario. JKHL is a travel agency that provides travel packages to worldwide destinations. JKHL Travel has offices in Washington, D.C., New York, Chicago, Mumbai, and Tokyo.

As JKHL grows, it encounters major limitations with its file-based integration infrastructure. In this chapter, we describe JKHL's integration requirements, the problems they encountered, and their plans to implement a managed file transfer solution: WebSphere MQ File Transfer Edition.

The topics that we discuss in this chapter are:

- 6.1, "Customer overview" on page 148
- 6.2, "Current integration solution" on page 150
- 6.3, "Proposed solution" on page 151
- 6.4, "After implementing an FTE solution" on page 154

6.1 Customer overview

In this section, we provide some background information about JKHL Travel (JKHL), which includes:

- Business profile and goals
- Existing environment
- Integration requirements

6.1.1 Business profile

JKHL currently has five offices worldwide: New York, Washington, Chicago, Mumbai, and Tokyo. All offices are responsible for selling JKHL travel packages. Each office has the following additional responsibilities:

- Washington: Worldwide management
- New York: Financial analysis, marketing, travel package development, and IT management.
- Chicago: Central data repository

As JKHL's business expanded from a single United States (U.S.) office to three U.S. and two overseas offices, the company encountered increasingly difficult problems using the file-based information integration.

Business goals

JKHL wants to expand their travel business while at the same time reducing their cost of operations. They see opportunities to reduce the amount of time that their staff spends on IT-related functions and more time on selling their products. In addition, they want to correct the serious IT infrastructure problems that are hampering their ability to manage and grow their business.

JKHL's major goals are to:

- Maintain secure operations
- Provide worldwide offices with standardized tools and reporting packages
- Reduce operational costs and time delays
- Reduce human error through automation
- Achieve accurate and efficient accounting of business operations
- Facilitate the development and marketing of new travel packages
- Provide the means to achieve a smooth transition when adding new offices

6.1.2 Existing environment

We must consider the existing environment at JKHL from a business perspective and an IT perspective. Figure 6-1 illustrates the JKHL offices.

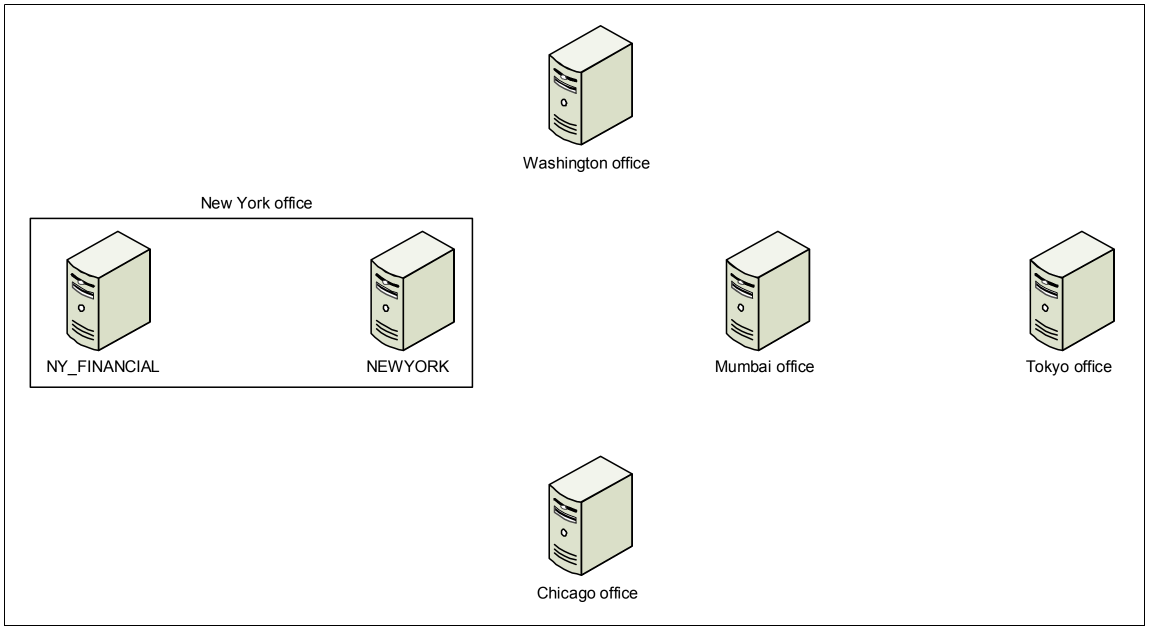


Figure 6-1 JKHL offices

Business perspective

JKHL Travel's business environment is organized around the following three activities:

- Setting up and maintaining its offices to support JKHL's business activities, which includes both brick and mortar and Internet sales.
- Financial reporting: Offices must provide daily sales and expense reports to the New York office where the NY Financial team consolidates and analyzes those reports, and then forwards the reports to the Chicago office.
- Rapid integration of new offices into their existing business systems

IT perspective

JKHL's IT environment consists of the following key features:

- Software distribution: The New York office distributes machine images and software updates to all offices. Distribution is accomplished using FTP and DVDs that are sent through the mail.

- Financial reports, destination reports, and proposed travel packages are received using a variety of means, which includes FTP and e-mail.
- WebSphere MQ infrastructure: To support an enterprise-packaged application, JKHL has an WebSphere MQ installed in all offices except the recently-acquired Tokyo office (where WebSphere MQ is not yet installed).
- Host machines and operating systems: The host machines that JKHL has are:
  - WASHINGTON, Windows XP Professional
  - NEWYORK, RedHat Linux
  - NY_FINANCIAL, Windows XP Professional
  - CHICAGO, z/OS
  - MUMBAI, RedHat Linux
  - TOKYO, RedHat Linux

6.1.3 Integration requirements

JKHL's file-based integration requirements are:

- To New York Financial: Receive ad hoc reports of changes to office system passwords. Size: 1 to 20 K.
- To New York: Receive daily financial reports from each JKHL office and update the Chicago corporate database. Size 10 to 50 K
- To all offices: Periodically distribute disk images and software updates from New York. Size: 1 to 10 GB
- To all offices: Periodically distribute new travel packages. Size: 1 to 10 MB.
- To all offices: Periodic package pricing update files are distributed from New York. Size: 10 to 50 K.

6.2 Current integration solution

JKHL's current integration approach evolves piecemeal as their organization grows. Methods that were initially used to integrate a local office are now being used to integrate a worldwide operation.

6.2.1 Methods used

Most of JKHL's data integration is accomplished through manual file transfer: FTP and, for large files, mailing DVDs.

Some examples are:

- Disk images and software updates are distributed using the mail.
- Daily financial reports, pricing updates, and office system passwords are manually transferred using e-mail, FTP, and mail.

6.2.2 Problems with the current methodology

JKKL encountered serious problems with its current file-based integration approach. These problems are serious enough to limit the success of their business:

- There is no central operational control and logging of file transfers
- Large file transfers are so unreliable that mailed DVDs must be used. This method requires a week or more to reach JKHL's overseas offices. Furthermore, the process of preparing these files for shipment is labor intensive. JKHL must:
  - Create DVD-sized segmented zip files of the disk image
  - Burn DVDs for each destination office.
  - Package the DVDs for each office.
  - Complete the required paperwork to send each package to the appropriate office.
  - On the receiving end, each office must copy the DVDs to a temporary drive, and then unzip the image.
- The limitations of the FTP approach that is used for most of JKHL's file transfers are:
  - No security
  - No logging
  - Data integrity problems
  - Routing over multiple hops is manual
  - There is no checkpoint restart
  - Each transfer ties up a port
  - Its manual nature makes it prone to human error
  - Does not provide automatic code page conversion

6.3 Proposed solution

It was proposed to JKHL Travel that they leverage their existing WebSphere MQ infrastructure by using the managed file transfer capabilities of WebSphere MQ File Transfer Edition (FTE).

6.3.1 How WebSphere MQ FTE addresses current problems

Using FTE over its existing WebSphere MQ infrastructure allows JKHL to gain the following key capabilities:

- Centralized control and monitoring
- Reliability
- Security
- Automation
- Support for any file size
- Time independence
- Consolidating enterprise traffic
- Integrating with non-MQ protocols

Centralized control and monitoring

All of JKHL's worldwide file transfers can be controlled and logged from its Washington office.

Reliability

Checkpoint-restart, checksum, and MQ's assured delivery provide for reliable file transfers. These features address the file transfer reliability problems that JKHL is plagued with.

Security

MQ can provide Secure Socket Layer (SSL) security for all file transfers. All transfers previously using FTP are now secure.

Automation

File transfers need not be manual. You can set up scheduled, triggered, and scripted file transfers using FTE, which saves JKHL the cost of the staff time that was previously required for manual file transfers and eliminates human errors.

In addition, FTE uses the MQ infrastructure to automatically route files over multiple hops, for example, using multiple machines without having to invoke separate transfers as is required with FTP.

Support for any file size

FTE supports the transfer of files of any size. The checkpoint-restart capability ensures that, in the event of a network or other failure, a large file transfer is resumed where it left off, which allows JKHL to distribute large files, such as disk images and software updates, with out the slow and time consuming use of the mail.

Time-independence

JKHL can invoke transfers independent of the availability of the network and destination machine. They can transfer files in an asynchronous manner.

Consolidating enterprise traffic

JKHL can now use their MQ infrastructure to support both their enterprise packaged application and file transfer without impacting the performance of their packaged application. They can also assign priorities to transfers.

In addition, messaging and multiple transfers can use the same port, which means that they do not tie up a port with one transfer like they do with FTP.

Integrating with non-MQ protocols

FTE provides the means for JKHL to integrate its file transfers with offices that do not yet have MQ installed, such as its Tokyo office. The Tokyo office supports FTP. FTE can centrally initiate and log file transfers that require an interface to other file transfer protocols.

6.3.2 Solution scenarios

JKHL plans to implement its FTE managed file transfer solution in three phases.

Phase 1: Basic file transfers

In this phase, we explore the basic capabilities and configuration of FTE, for intended use with JKHL solutions. We describe this phase in Chapter 7, “Phase 1: Basic file transfers” on page 159.

Phase 2: Multi-step transfers

In this phase, FTE is applied to the development of JKHL’s file-based financial reporting system. We describe this phase in Chapter 8, “Phase 2: Multi-step transfers” on page 191.

Phase 3: Complex transfers

In this phase, FTE is applied to the development of JKHL’s file-based travel package distribution system, the completion of the financial reporting system, and the development of JKHL’s pricing update system. We describe this phase in Chapter 9, “Phase 3: Complex transfers” on page 227.

6.4 After implementing an FTE solution

The bottom-line reason for implementing JKHL's FTE managed file transfer solution is to achieve the following business goals:

- Maintain secure operations
- Provide worldwide offices with uniform corporate tools and reporting packages
- Reduce operational costs, time delays, and human error
- Achieve accurate and efficient accounting of business operations
- Facilitate the development and marketing of new travel packages
- Provide the means to achieve a smooth transition when adding new offices.

In this section, we discuss how, during each of the three development phases, each of JKHL's business goals were met with an FTE solution.

6.4.1 Maintaining secure operations

In this section, we describe the WebSphere MQ File Transfer Edition solutions that address this business goal.

Phase 1

The business goals that are addressed for this phase are:

- Established Secure Socket Layer security (SSL) channels between all JKHL offices with FTE server: WASHINGTON, NEWYORK, CHICAGO, AND MUMBAI. This security is an out-of-the-box MQ option. (7.4, "Enabling SSL security on FTE client agents" on page 171)
- Demonstrated a secure file transfer between WASHINGTON and NEWYORK (both are FTE servers). (7.2, "Basic secure file transfers with optional parameters" on page 162)
- Established an SSL security channel to the JKHL office with an FTE client.(7.4, "Enabling SSL security on FTE client agents" on page 171)
- Demonstrated a secure transfer between MUMBAI, which is an FTE server and NY_FINANCIAL, which is an FTE client. (7.5, "Multi-hop file transfers" on page 180)

Phase 3

The business goals that are addressed for this phase are:

- Provided a secure mechanism for FTE-enabled offices to exchange files with their non-FTE-enabled office, TOKYO, using the Secure Shell (SSH2) protocol. (9.3.1, "Retrieving TOYKO daily financial reports using SFTP" on page 253).

6.4.2 Providing worldwide offices with standardized software tools and reporting packages

In this section, we describe the WebSphere MQ File Transfer Edition solutions that address this business goal.

Phase 1

The business goals that are addressed for this phase are:

- Enabled the reliable electronic transfer of large software packages to FTE-enabled offices.
- Demonstrated the checkpoint restart capability of FTE by transferring a 1 GB software distribution file between the NEWYORK and MUMBAI offices. The network connection was broken during the transfer, and then restored (7.7, "Checkpoint restart" on page 186).

6.4.3 Reducing operational costs, time delays, and human error

The WebSphere MQ File Transfer Edition solutions addressing this business goal are described here.

Phase 1

The business goals that are addressed for this phase are:

- Demonstrated the use of FTE priorities when transferring files. The transfer of a 1 GB large software distribution file was started between WASHINGTON and NEWYORK. A high priority, small travel package update was then sent from NEWYORK to WASHINGTON. It immediately reached its destination (7.3, "Using the transfer priorities" on page 168).
- Demonstrated FTE's ability to automatically provide multi-hop transfers by transferring a file between the MUMBAI and NY_FINANCIAL office. The file had to traverse the NEWYORK office to get to NY_FINANCIAL (7.5, "Multi-hop file transfers" on page 180).

- Established MQ channel compression to automatically compress file data that is sent using FTE thereby reducing the time required for transmission (7.6.3, “Configuring WebSphere MQ compression” on page 183).
- Eliminated the time-consuming, costly, and unreliable process of having to burn DVDs and mail software distributions to their worldwide offices. Demonstrated the checkpoint restart capability of FTE by transferring a 1 GB software distribution file between the NEWYORK and MUMBAI offices. The network connection was broken during the transfer then restored (7.7, “Checkpoint restart” on page 186).

Phase 2

The business goals that are addressed for this phase are:

- Eliminated the previous error-prone system of manually reporting through e-mail, FTP, and mail.
- Completely automated the receipt and processing of daily financial reporting form all FTE-enabled offices (Chapter 8, “Phase 2: Multi-step transfers” on page 191).

Phase 3

The business goals that are addressed for this phase are:

- Automated the receipt, through SFTP, of daily financial reports from JKHL’s non-FTE-enabled office, TOKYO (9.3.1, “Retrieving TOYKO daily financial reports using SFTP” on page 253).
- Greatly reduced errors and dramatically reduced the processing time by automating the backend portion of the financial reporting system that updates JKHL’s CHICAGO database (9.3, “Completing the financial reporting system” on page 253).
- Automated the distribution of new travel packages (9.2, “Distributing ad hoc travel package updates” on page 229).
- Eliminated the previous error-prone system of manually reporting through e-mail, FTP, and mail. Automated the pricing update system, including, data transformation, and a database update (9.4, “Receiving and distributing pricing updates” on page 257).

6.4.4 Achieving accurate and efficient accounting of business operations

In this section, we describe the WebSphere MQ File Transfer Edition solutions that address this business goal.

Phase 2

The business goals that are addressed for this phase are:

- Completely replaced the existing inefficient manual system.
- Automated the receipt and processing of daily financial reporting form all FTE-enabled offices (Chapter 8, “Phase 2: Multi-step transfers” on page 191).

Phase 3

The business goals that are addressed for this phase are:

- Achieved greater accuracy and efficiency by automating the back end portion of the financial reporting system that updates JKHL’s CHICAGO database (9.3, “Completing the financial reporting system” on page 253).
- Automated the pricing update system, including data transformation and a database update (9.4, “Receiving and distributing pricing updates” on page 257).

6.4.5 Facilitating the development and marketing of new travel packages

In this section, we describe the WebSphere MQ File Transfer Edition solution that addresses this business goal.

Phase 3

The business goals that are addressed for this phase are:

- Eliminated the previous manual and error-prone system of using e-mail, FTP, and mail.
- Automated the distribution of new travel packages (9.2, “Distributing ad hoc travel package updates” on page 229).

6.4.6 Achieving a smooth transition when adding new offices

In this section, we describe the WebSphere MQ File Transfer Edition solutions that address this business goal.

Phase 3

Demonstrated fteAnt-cURL’s ability to access the SFTP protocol that is required by the newly-acquired TOKYO office. The TOKYO office currently uses SFTP. fteAnt-cURL provides a transitional solution that can be used until the TOKYO office is FTE enabled. This same approach can be used to access a wide variety of other protocols that might be used, in transition, by future non-FTE-enabled offices. The available protocols include: FTP, FTPS, SFTP, TFTP, HTTP, HTTPS, SCP, TELNET, and DICT (9.3.1, “Retrieving TOYKO daily financial reports using SFTP” on page 253).

7

Phase 1: Basic file transfers

In this chapter, we describe the JKHL basic file transfer scenarios, which include simple point-to-point file transfers with parameters, directory transfers, priority transfers, and multi-hop transfers. We also introduce functions, such as using SSL, compression, and checkpoints. For all scenarios, we provide steps for WebSphere MQ File Transfer Edition.

The chapter contains the following scenarios:

- Overview of JKHL scenarios
- Basic file transfers with optional parameters
- Enabling SSL security on client agents
- Setting channel compression for large file transfers
- Checkpoint restart

The topics that we discuss in this chapter are:

In this chapter, we use the fictional company scenario of JKHL Travel, which we described in Chapter 6, “The business scenario that we use in this book” on page 147.

7.1 Scenario overview

JKHL has many business processes that use file transfers between their five offices, as shown in Figure 7-1. JKHL plans to implement a managed file transfer solution of IBM WebSphere MQ File Transfer Edition in three phases. We describe phase 1 in this chapter, where JKHL creates and tests a basic WebSphere MQ File Transfer Edition topology.

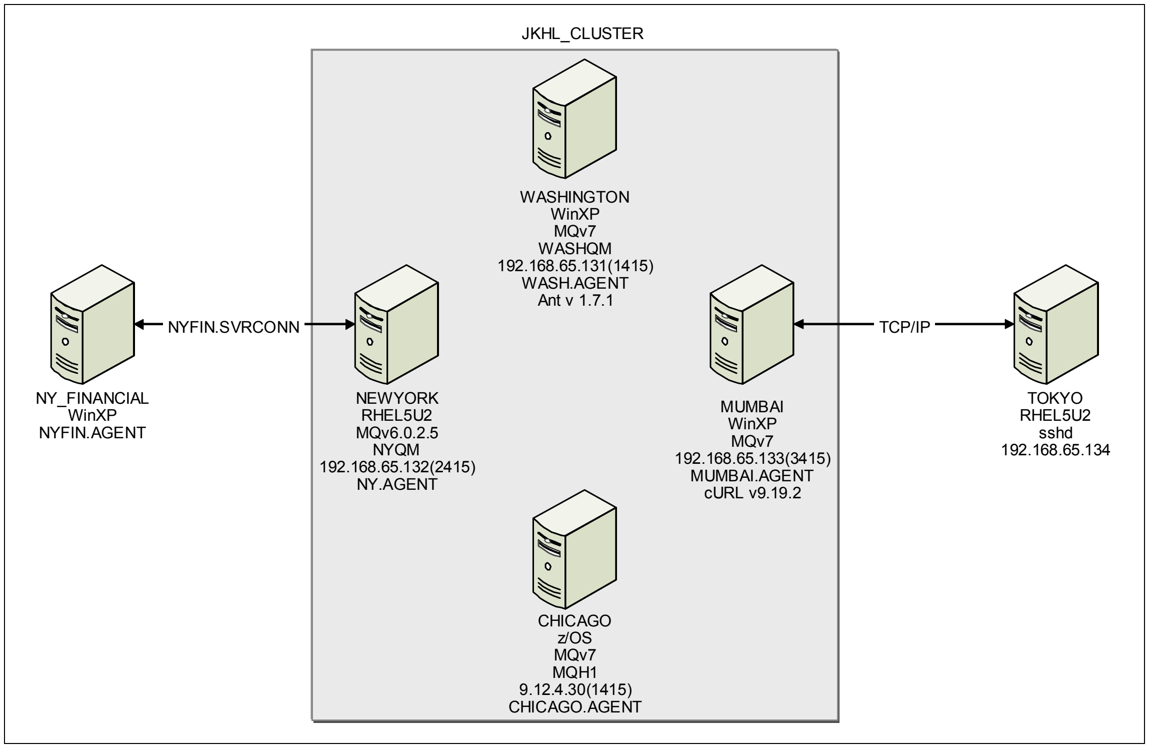


Figure 7-1 JKHL file transfer architecture

It is important for JKHL to complete basic secure file transfers between six offices, for instance, new travel packages are transferred between Washington and New York.

JKHL hopes to use the FTE graphical user interface to submit file transfers and configure different transfer options, such as transferring a directory, using the MD5 checksum after finishing file transfers, and setting different priorities during concurrent file transfers.

Multi-hop transfers often occur between remote offices, such as Mumbai and the New York Financial.

Another key point in the solution is that there are no IT employees in JKHL remote offices, which means that simple and easy-use client software must be installed and configured in these offices. At the same time, JKHL headquarters must ensure that every FTE client agent has authentication for security in remote offices. The FTE client agent must be securely connected, through the WebSphere MQ SSL protocol, to a queue manager on an FTE server.

In addition, performance is also a point that JKHL is considering. The MQ compression mechanism can help JKHL reduce the bandwidth usage when log or text files are transferred.

JKHL's basic requirements are satisfied in the following scenarios:

- Basic secure file transfers with optional parameters.
- File transfers with different priorities.
- Enabling SSL security on FTE client agents.
- Multi-hop file transfers.
- Setting MQ channel compression for large text file transfers.
- File transfer recovery.

There is an assumption before JKHL begins these scenarios. In Figure 7-1 on page 161, cluster sender and receiver channels are used in four offices, including Washington, New York, Chicago, and Mumbai. These channels enabled WebSphere MQ SSL configuration. Because MQ SSL enablement between queue managers belongs to the WebSphere MQ topic, we do not describe these SSL configurations. These configurations include normal sender and receiver channels and cluster sender and receiver channels. MQ SSL configuration is located at the WebSphere MQ Information Center:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzas.doc/sy11550_.htm

You can also refer to the IBM Redpaper publication, *WebSphere MQ V6, WebSphere Message Broker V6, and SSL*, REDP-4140.

7.2 Basic secure file transfers with optional parameters

In the past, JKHL shipped DVDs for large disk images and software updates. For smaller files, such as financial reports and pricing update files, JKHL often uploaded and downloaded them through FTP. JKHL decides to replace slow and non-reliable transfer modes with IBM WebSphere MQ File Edition.

7.2.1 The requirement

JKHL plans to set up a basic secure file transfer system between all offices in phase 1.

Distributing new travel packages to FTE-enabled offices

FTE can easily complete the basic delivery to the WASHINGTON, NEW YORK, CHICAGO, MUMBAI, and NY_FINANCIAL offices. All of these offices are FTE-enabled.

Providing secure and reliable file transfers

The financial reports and pricing update files are confidential. JKHL hopes that FTE can provide secure and reliable transfers to change the past delivery methods: no security and no reliability.

7.2.2 The solution

FTE provides secure, reliable, and flexible file transfers and helps JKHL build a basic transfer system for all offices. JKHL can specify parameters, such as MD5 checksum, overwrite files on the destination system, transfer directory including subdirectories, and binary or text transfer before file transfers are started.

7.2.3 Starting a new file transfer with WebSphere MQ Explorer

In this scenario, a directory of new travel package files are sent from Washington to New York by SSL channels. JKHL uses the FTE plug-in for WebSphere MQ Explorer to create this transfer with MD5 checksum.

To test the scenario:

1. On the Washington machine, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click **WebSphere MQ Explorer** to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **WASHQM**.
3. To launch the wizard as shown in Figure 7-2 on page 164, right-click **Transfer Templates**, and select **New Transfer**.

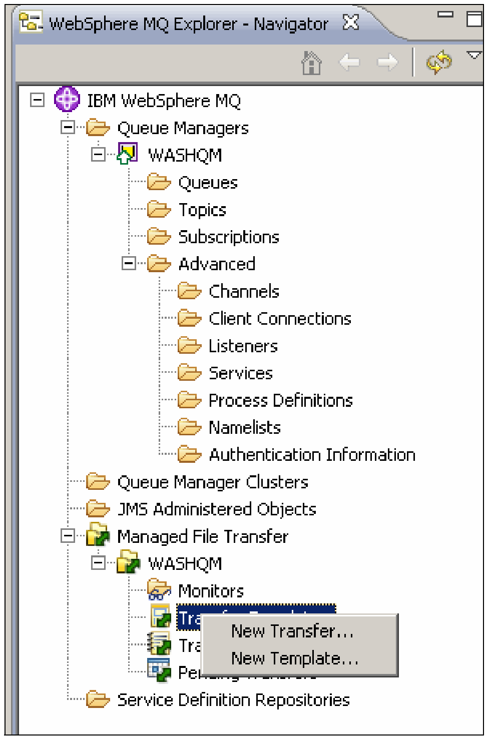
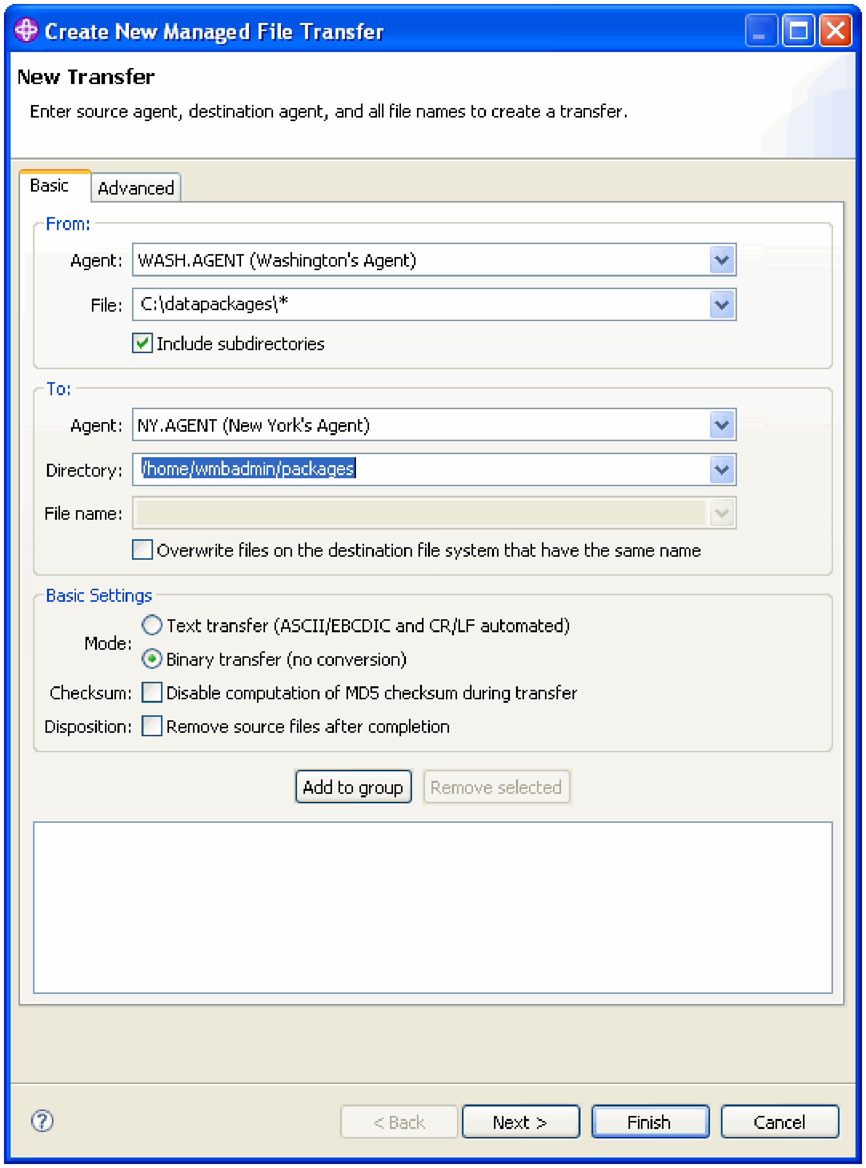


Figure 7-2 Opening a new file transfer request

4. In the Creating New Managed File Transfer wizard, shown in Figure 7-3 on page 165, perform the following actions:
   a. Select source agent name, **WASH.AGENT**, in the agent list.
   b. Specify the directory, including new travel package files, C:\newpackages\*, on the source agent.
   c. Enable the **Include subdirectories** option.
   d. Select destination agent name, **NY.AGENT,** in the agent list.
   e. Specify the receiving directory, /home/wmbadmin/packages, on the destination agent.
   f. In the **Basic Settings** option, choose **Binary transfer** mode.
   g. Ensure that **Checksum** is not selected.


Create New Managed File Transfer
New Transfer
Enter source agent, destination agent, and all file names to create a transfer.
Basic
Advanced

From:

Agent: WASH.AGENT (Washington's Agent)

File: C:\datapackages\*

[✓] Include subdirectories

To:

Agent: NY.AGENT (New York's Agent)

Directory: /home/wmbadmin/packages

File name:

[ ] Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ( ) Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ Binary transfer (no conversion)]

Checksum: [ ] Disable computation of MD5 checksum during transfer

Disposition: [ ] Remove source files after completion

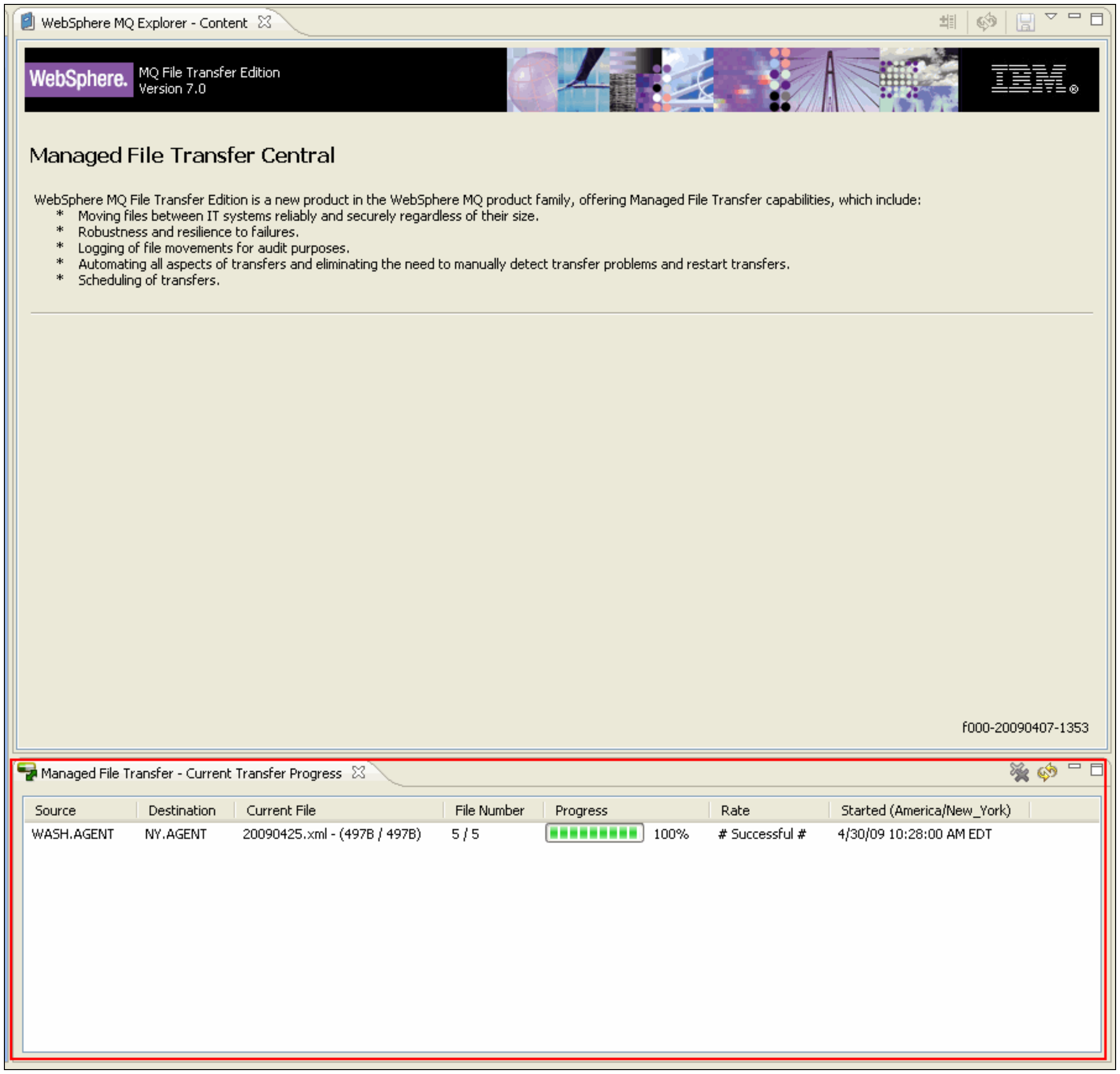
Figure 7-3 Creating a basic file transfer

h. Click **Finish.** Now a secure transfer from Washington to New York is started.

7.2.4 Viewing the transfer progress

JKHL wants to view transfer progress in the WebSphere MQ Explorer:

1. In WebSphere MQ Explorer, look at the view **Current Transfer Progress**, as shown in Figure 7-4.


WebSphere MQ Explorer - Content
WebSphere. MQ File Transfer Edition Version 7.0
IBM

Managed File Transfer Central

WebSphere MQ File Transfer Edition is a new product in the WebSphere MQ product family, offering Managed File Transfer capabilities, which include:
* Moving files between IT systems reliably and securely regardless of their size.
* Robustness and resilience to failures.
* Logging of file movements for audit purposes.
* Automating all aspects of transfers and eliminating the need to manually detect transfer problems and restart transfers.
* Scheduling of transfers.

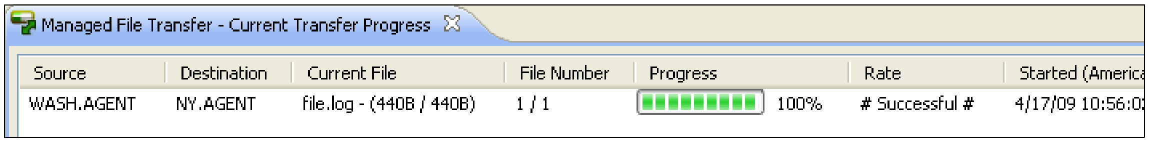
f000-20090407-1353

Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress	Rate	Started (America/New_York)
WASH.AGENT	NY.AGENT	20090425.xml - (497B / 497B)	5 / 5	100%	# Successful #	4/30/09 10:28:00 AM EDT

Figure 7-4   Managed File Transfer: Current Transfer Progress

2. In the Current Transfer Progress view, examine the file transfer progress and results, as shown in Figure 7-5. The information includes source, destination, current file name, file number, progress state, transfer rate, and started time.


Managed File Transfer - Current Transfer Progress

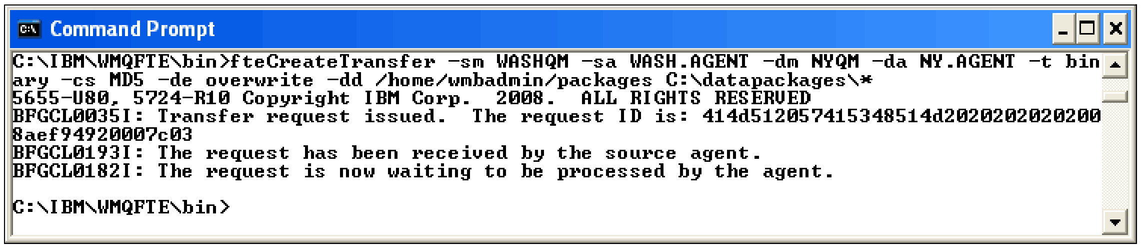
Source	Destination	Current File	File Number	Progress	Rate	Started (Americ
WASH.AGENT	NY.AGENT	file.log - (440B / 440B)	1 / 1	100%	# Successful #	4/17/09 10:56:0

Figure 7-5 Transfer logs in Managed File Transfer: Current Transfer Progress view

7.2.5 Creating a new file transfer with the command line

JKHL also creates a transfer with the command line on Windows and performs the following actions:

1. Open the command line and switch paths to the bin directory in the FTE installation path, as shown in Figure 7-6. Input the command content (shown in Example 7-1), and press the **Enter**.



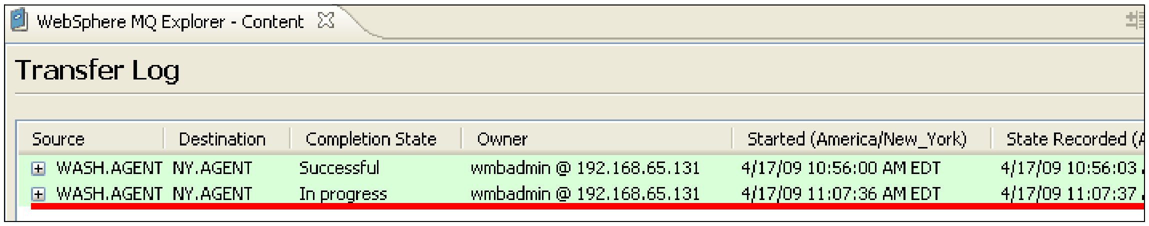
```
C:\IBM\WMQFTE\bin>fteCreateTransfer -sm WASHQM -sa WASH.AGENT -dm NYQM -da NY.AGENT -t bin
ary -cs MD5 -de overwrite -dd /home/wmbadmin/packages C:\datapackages\*
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0035I: Transfer request issued.  The request ID is: 414d51205741534851 4d2020202020200
8aef94920007c03
BFGCL0193I: The request has been received by the source agent.
BFGCL0182I: The request is now waiting to be processed by the agent.

C:\IBM\WMQFTE\bin>
```

Figure 7-6 The command display of file transfer from Washington to New York

Example 7-1 The command content of file transfer from Washington to New York

```
fteCreateTransfer
     -sm WASHQM
     -sa WASH.AGENT
     -dm NYQM
     -da NY.AGENT
     -t binary
     -cs MD5
     -de overwrite
     -r -dd /home/wmbadmin/packages
     “C:\datapackages\*”
```

2. Check the transfer result, as shown in Figure 7-7 on page 168.


WebSphere MQ Explorer - Content

Transfer Log

Source	Destination	Completion State	Owner	Started (America/New_York)	State Recorded (A
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/17/09 10:56:00 AM EDT	4/17/09 10:56:03
WASH.AGENT	NY.AGENT	In progress	wmbadmin @ 192.168.65.131	4/17/09 11:07:36 AM EDT	4/17/09 11:07:37

Figure 7-7 Current transfer log for the command line

Now, the scenario is complete and successful. JKHL built a basic secure file transfer between two offices.

7.3 Using the transfer priorities

JKHL sent a lot of package update files and financial report files on the WebSphere MQ FTE network; however, these files are sent under the same priority level. JKHL will meet a new requirement.

7.3.1 The requirement

JKHL has important pricing update files to transfer urgently from New York to other offices. The priority of these files is high. JKHL hopes FTE can provide the priority aware transfer services.

7.3.2 The solution

WebSphere MQ File Transfer Edition can implement low or high priority transfers. The priority is based on WebSphere MQ from 0 to 9. JKHL can use the lowest priority to transfer some unimportant files and issue the high priority transfer for urgent files.

7.3.3 Starting file transfers with different priorities

JKHL decides to use the file priorities in special conditions. JKHL is concerned that the transfer of large low priority software update files are blocking the way for the transfer of small higher priority pricing update files. For this scenario, a 1 GB large file first is submitted to FTE with a low priority, and then some small files in one directory are transferred with a high priority.

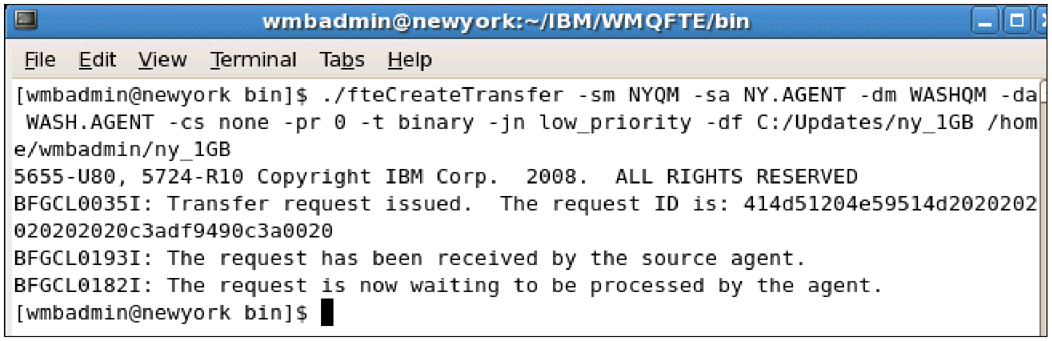
To build the file transfer:

1. Open a terminal window on the New York Linux machine, input the transfer command with a low priority, shown in Example 7-2, for the 1 GB large software update file transfer, and press **Enter**.

Example 7-2 The transfer request command for 1 GB file

```
fteCreateTransfer
      sm NYQM
      -sa NY.AGENT
      -dm WASHQM
      -da WASH.AGENT
      -cs none
      -pr 0
      -t binary
      -jn low_priority
      -df C:\Updates\ny_1GB
      /home/wmbadmin/ny_1GB
```

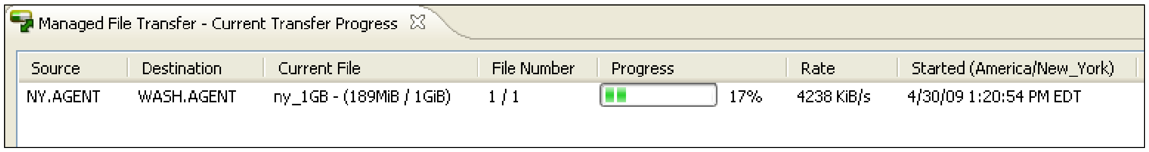
The **fteCreateTransfer** command with low priority is issued successfully, as seen in Figure 7-8.


wmbadmin@newyork:~/IBM/WMQFTE/bin
File Edit View Terminal Tabs Help

```
[wmbadmin@newyork bin]$ ./fteCreateTransfer -sm NYQM -sa NY.AGENT -dm WASHQM -da
 WASH.AGENT -cs none -pr 0 -t binary -jn low_priority -df C:/Updates/ny_1GB /hom
e/wmbadmin/ny_1GB
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0035I: Transfer request issued.  The request ID is: 414d51204e59514d2020202
020202020c3adf9490c3a0020
BFGCL0193I: The request has been received by the source agent.
BFGCL0182I: The request is now waiting to be processed by the agent.
[wmbadmin@newyork bin]$
```

Figure 7-8 The low priority transfer is issued

2. On the Washington machine, view the 1 GB file transfer progress in the WebSphere MQ Explorer, as shown in Figure 7-9.



Source	Destination	Current File	File Number	Progress		Rate	Started (America/New_York)
NY.AGENT	WASH.AGENT	ny_1GB - (189MiB / 1GiB)	1 / 1	🟩	17%	4238 KiB/s	4/30/09 1:20:54 PM EDT

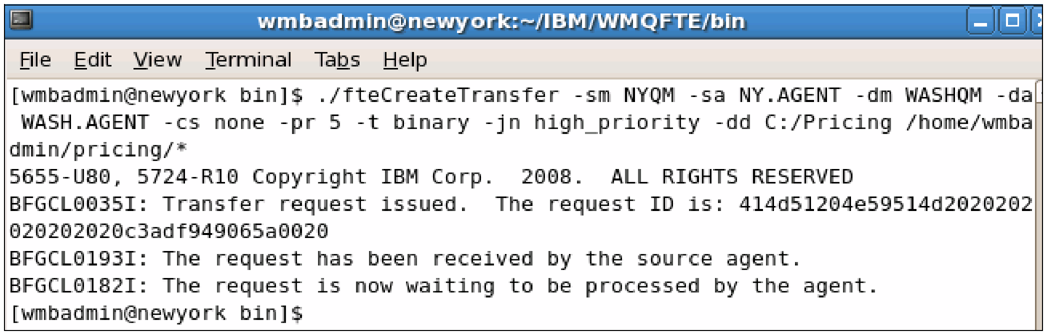
Figure 7-9 The low priority transfer progress

3. Input the urgent transfer command for small files with high priorities, as shown in Example 7-3. The high priority is 5.

Example 7-3 The command of creating a high priority transfer

```
fteCreateTransfer
      -sm NYQM
      -sa NY.AGENT
      -dm WASHQM
      -da WASH.AGENT
      -cs none
      -pr 5
      -t binary
      -jn high_priority
      -dd C:\Pricing\
      /home/wmbadmin/pricing/*
```

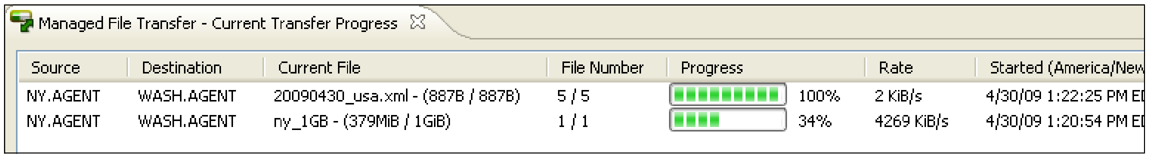
The **fteCreateTransfer** command with high priority is issued successfully, as shown in Figure 7-10.


wmbadmin@newyork:~/IBM/WMQFTE/bin
File Edit View Terminal Tabs Help

```
[wmbadmin@newyork bin]$ ./fteCreateTransfer -sm NYQM -sa NY.AGENT -dm WASHQM -da
 WASH.AGENT -cs none -pr 5 -t binary -jn high_priority -dd C:/Pricing /home/wmba
dmin/pricing/*
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0035I: Transfer request issued.  The request ID is: 414d51204e59514d2020202
020202020c3adf949065a0020
BFGCL0193I: The request has been received by the source agent.
BFGCL0182I: The request is now waiting to be processed by the agent.
[wmbadmin@newyork bin]$
```

Figure 7-10 The high priority transfer is issued

4. In the Current Transfer Progress view, observe the file transfer progress. The small files with high priority finished transferring, and the 1 GB large file with low priority is sent from New York to Washington, as shown in Figure 7-11.



Source	Destination	Current File	File Number	Progress		Rate	Started (America/New
NY.AGENT	WASH.AGENT	20090430_usa.xml - (887B / 887B)	5 / 5	▮▮▮▮▮▮▮▮▮▮	100%	2 KiB/s	4/30/09 1:22:25 PM E
NY.AGENT	WASH.AGENT	ny_1GB - (379MiB / 1GiB)	1 / 1	▮▮▮▮	34%	4269 KiB/s	4/30/09 1:20:54 PM E

Figure 7-11 The high priority transfer progress

7.4 Enabling SSL security on FTE client agents

WebSphere MQ File Transfer Edition enables SSL on WebSphere MQ channels, including cluster channels and client channels.

7.4.1 The requirement

JKHL configured SSL properties between Washington, New York, Mumbai, and Chicago. These offices use cluster sender and receiver channels on queue managers.

However JKHL has some small offices, such as the New York financial office. In these offices, JKHL has no IT administrator for the file transfer platform. It means queue managers cannot be configured. Only FTE client agents are allowed to be installed.

7.4.2 The solution

JKHL's NewYork financial office can access remote queue managers on the New York FTE server using WebSphere MQ SSL client channels for security authentications.

7.4.3 Setting up SSL security on the FTE client agent

In this scenario, we set up a one-way SSL channel connection. One-way means that only the queue manager presents a certificate, which the client authenticates.

There are a number of ways to obtain a certificate for queue manager:

- Create self-signed certificates.
- Have an internal certification authority.
- Request a certificate from a certification authority.

Self-signed certificates are used in the WebSphere MQ server-connection channel from NYFIN to New York and created by the WebSphere MQ iKeyMan tool.

To create a key repository for queue manager NYQM on Linux:

1. On the New York machine, confirm that the gsk7bas package is installed.
2. Open a Linux terminal, and issue commands by the wmbadmin user.

3. Create a key repository for the queue manager using the command shown in Example 7-4:
   - Key database type: CMS
   - File Name: key.kdb
   - Location: /var/mqm/qmgrs/NYQM/ssl
   - Password: 111111

Example 7-4 The command of creating a key repository

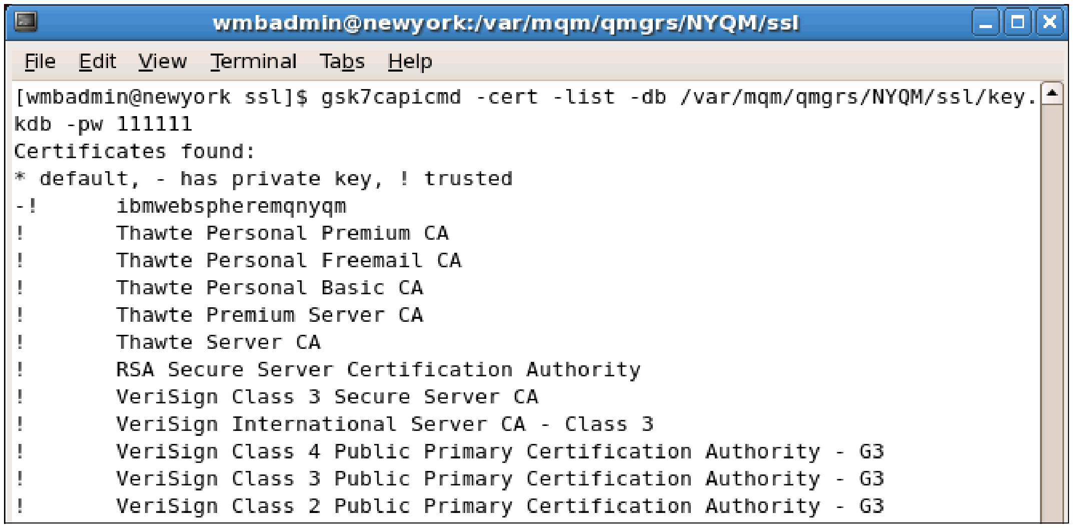
```
gsk7capicmd -keydb -create -db /var/mqm/qmgrs/NYQM/ssl/key.kdb -pw
   111111 -type cms -expire 365 -stash
```

4. Create a self-signed certificate for the client using the command in Example 7-5:
   - Key Label: ibmwebspheremqnyqm (must be ibmwebspheremq followed by the queue manager name in lowercase)
   - Common Name: NYQM (You can have a different naming convention for the common name; feel free to enter any other value.)
   - Organization: JKHL (any company's name)

Example 7-5 The command of creating a self-signed certificate

```
gsk7capicmd -cert -create -db /var/mqm/qmgrs/NYQM/ssl/key.kdb -pw
   111111 -label ibmwebspheremqnyqm -dn "CN=NYQM,O=JKHL" -size 1024
   -x509version 3 -expire 365
```

5. List the certificates in the key repository using the command in Figure 7-12 on page 173.


wmbadmin@newyork:/var/mqm/qmgrs/NYQM/ssl
File Edit View Terminal Tabs Help

```
[wmbadmin@newyork ssl]$ gsk7capicmd -cert -list -db /var/mqm/qmgrs/NYQM/ssl/key.
kdb -pw 111111
Certificates found:
* default, - has private key, ! trusted
-!      ibmwebspheremqnyqm
!       Thawte Personal Premium CA
!       Thawte Personal Freemail CA
!       Thawte Personal Basic CA
!       Thawte Premium Server CA
!       Thawte Server CA
!       RSA Secure Server Certification Authority
!       VeriSign Class 3 Secure Server CA
!       VeriSign International Server CA - Class 3
!       VeriSign Class 4 Public Primary Certification Authority - G3
!       VeriSign Class 3 Public Primary Certification Authority - G3
!       VeriSign Class 2 Public Primary Certification Authority - G3
```

Figure 7-12 The certificates in the key repository

Now the NYQM has a certificate. NYQM presents this certificate to the NYFIN FTE client agent when the agent connects. To validate the queue manager's certificate, the client needs the certification authority (CA) certificate.

To extract the CA certificate on NYQM and copy it to NYFIN client agent:

6. Issue the extract command in Example 7-6), export the certificate to the local file system, and save the certificate as ibmwebspheremqnyqm.der:
   - File Name: ibmwebspheremqnyqm.der (binary format)
   - Location: /var/mqm/qmgrs/NYQM/ssl/

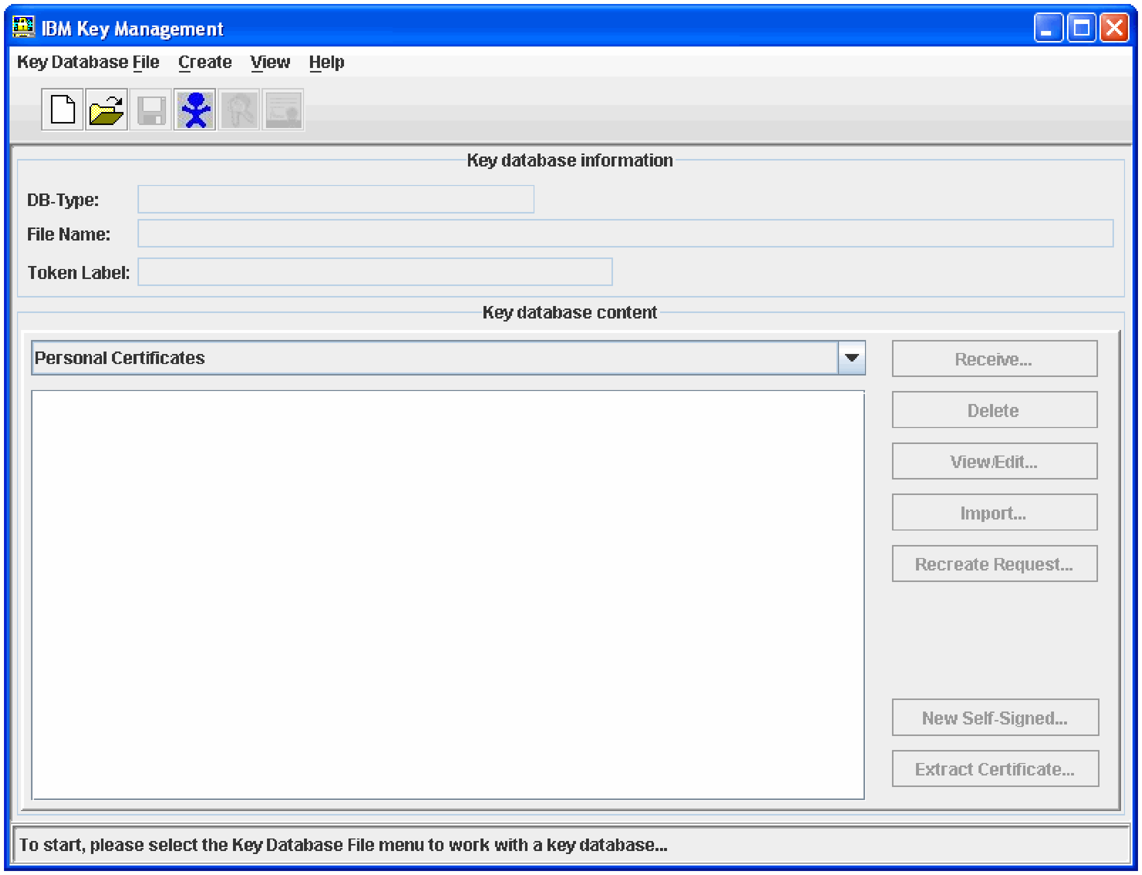
Example 7-6 Extracting the certificate to local file system

```
gsk7capicmd -cert -extract -db /var/mqm/qmgrs/NYQM/ssl/key.kdb -pw
   111111 -label ibmwebspheremqnyqm -target
   /var/mqm/qmgrs/NYQM/ssl/ibmwebspheremqnyqm.der -format binary
```

7. Copy the certificate file to the file system on NYFIN machine.

   To install the CA certificate in the FTE client agent's key repository, some steps are executed on the NYFIN machine:

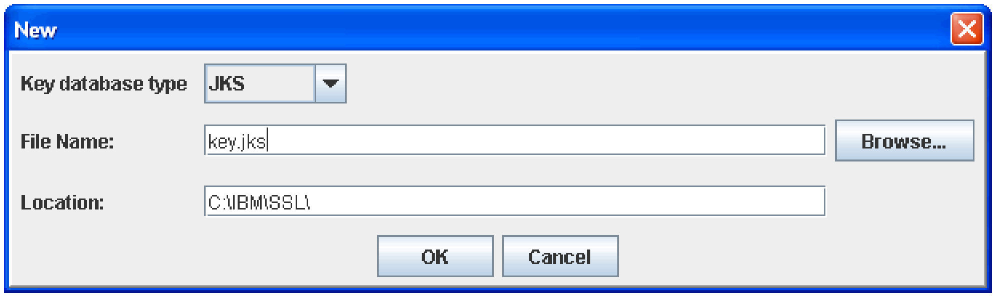
8. Put the CA certificate file (ibmwebspheremqnyqm.der) in a local directory, such as the C:\IBM\SLL directory on the NYFIN machine.
9. On the NYFIN machine, open the command prompt window, and switch to the <WMQFTE_install_directory>\jre\bin directory. Enter **ikeyman**, and open the IBM Key Management tool, as shown in Figure 7-13 on page 174.



To start, please select the Key Database File menu to work with a key database...

Figure 7-13 IBM Key Management Tool

10. Create a key repository for the FTE client agent on NYFIN. Select **Key Database File** → **New**, as shown in Figure 7-14 on page 175. Input the information and click **OK**:
    - Key database type: JKS (Java Applications must use the type)
    - File Name: key.jks
    - Location: C:\IBM\SSL



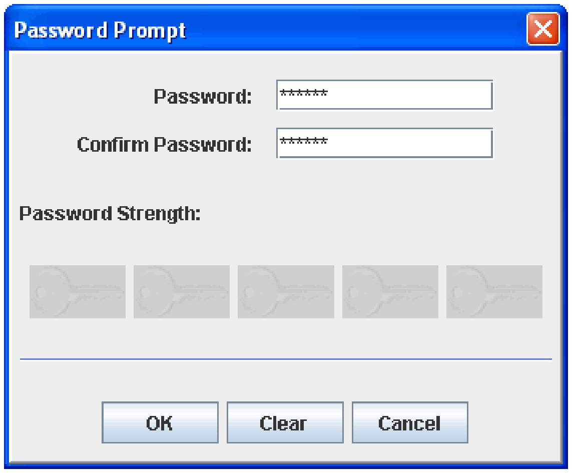
Key database type: JKS

File Name: key.jks

Location: C:\IBM\SSL\

Figure 7-14 Creating a key repository for NYFIN FTE client agent

11. At the password prompt, type the password for this repository, and then click **OK**, as shown in Figure 7-15.

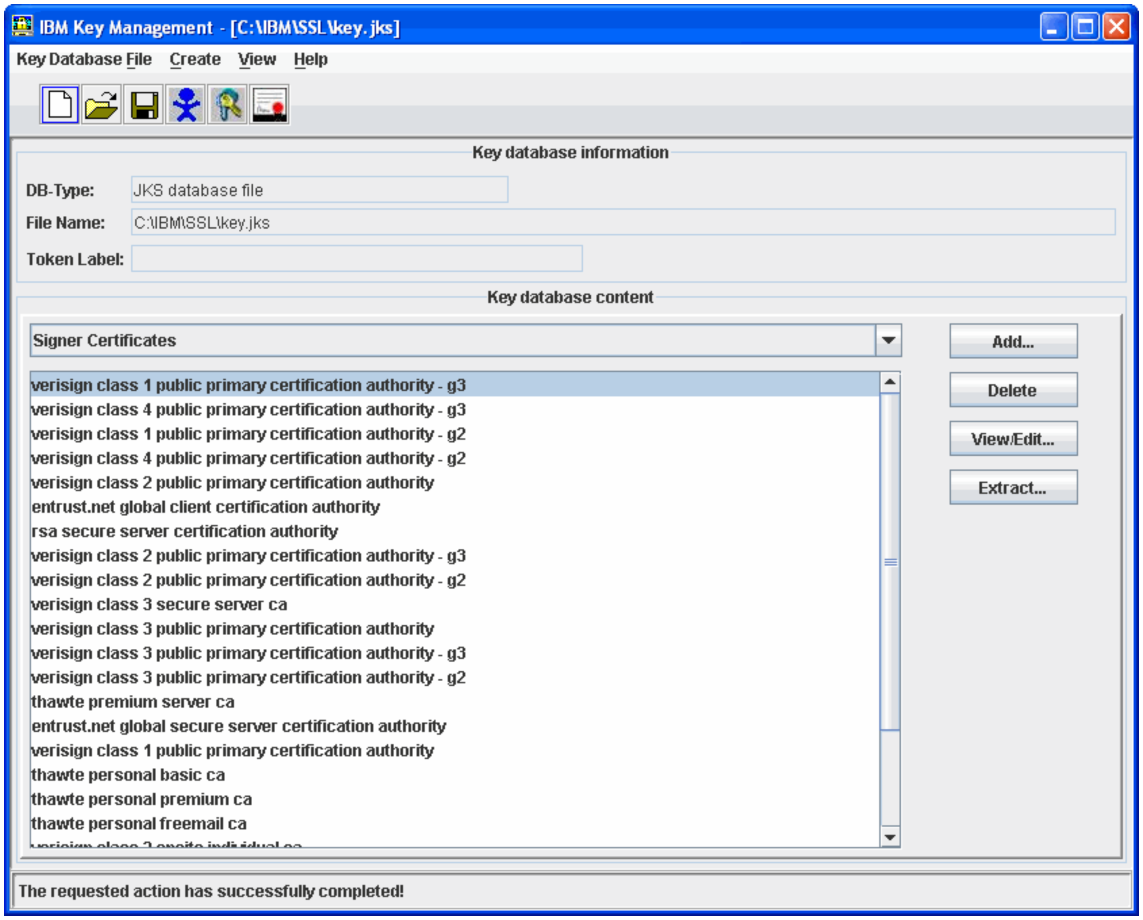


Password: *******

Confirm Password: *******

Figure 7-15 The password window

12. In the main view of the IBM Key Management Tool, click **Add**, as shown in Figure 7-16 on page 176.


IBM Key Management - [C:\IBM\SSL\key.jks]

Key database information

DB-Type: JKS database file

File Name: C:\IBM\SSL\key.jks

Token Label:

Key database content

Signer Certificates

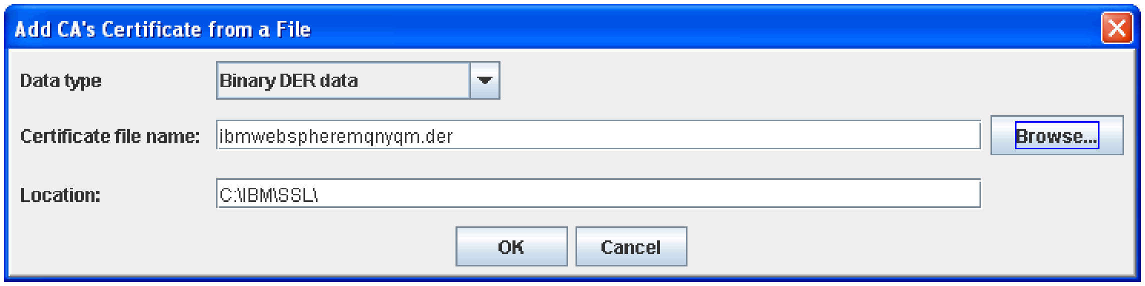
verisign class 1 public primary certification authority - g3
verisign class 4 public primary certification authority - g3
verisign class 1 public primary certification authority - g2
verisign class 4 public primary certification authority - g2
verisign class 2 public primary certification authority
entrust.net global client certification authority
rsa secure server certification authority
verisign class 2 public primary certification authority - g3
verisign class 2 public primary certification authority - g2
verisign class 3 secure server ca
verisign class 3 public primary certification authority
verisign class 3 public primary certification authority - g3
verisign class 3 public primary certification authority - g2
thawte premium server ca
entrust.net global secure server certification authority
verisign class 1 public primary certification authority
thawte personal basic ca
thawte personal premium ca
thawte personal freemail ca

The requested action has successfully completed!

Figure 7-16 The interface of IBM Key Management Tool

13. In the next window, Figure 7-17 on page 177, enter:
    - Certificate file name: ibmwebspheremqnyqm.der
    - Location: C:\IBM\SSL\

    Click **OK**.

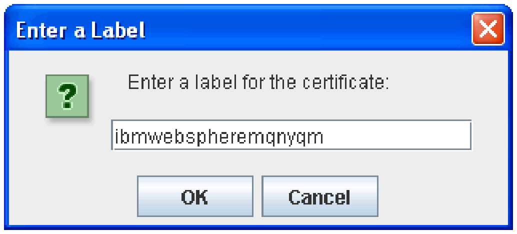

Data type: Binary DER data

Certificate file name: ibmwebspheremqnyqm.der

Location: C:\IBM\SSL\

Figure 7-17 Add the certificate to the key repository on NYFIN FTE client agent

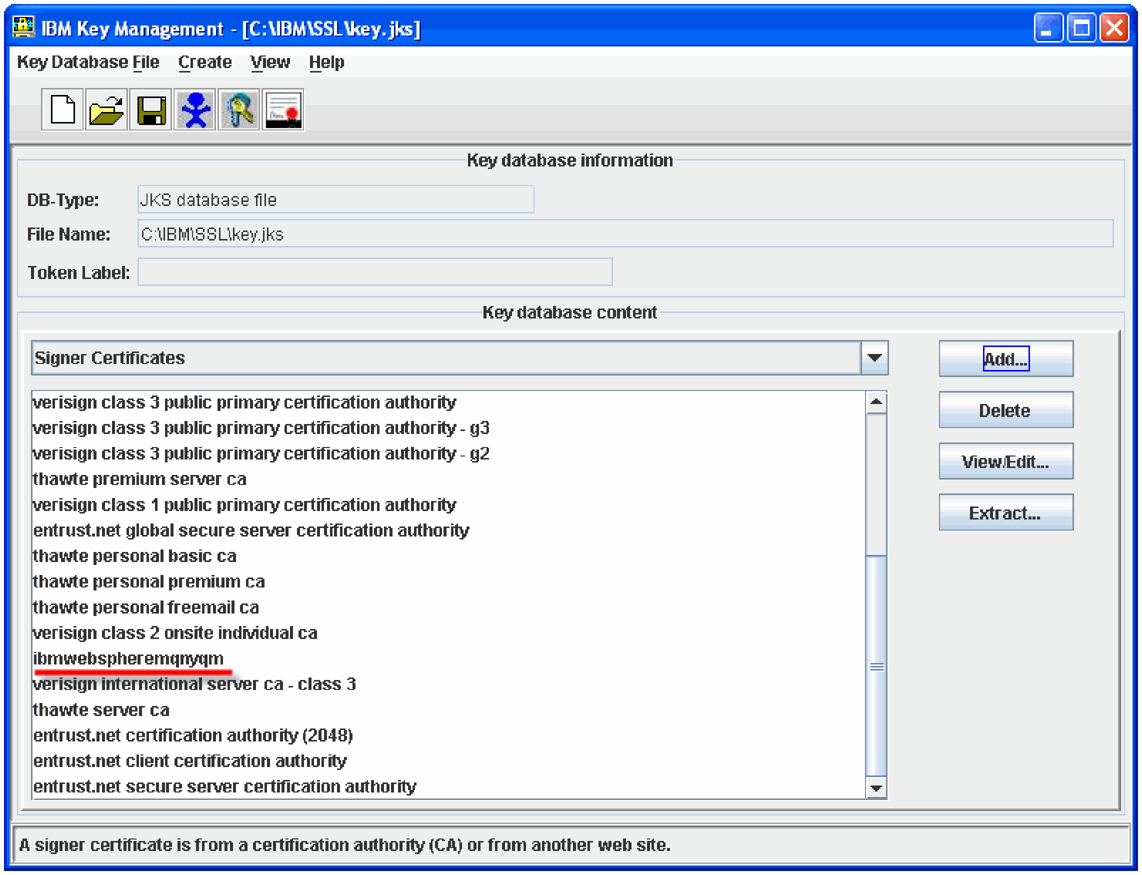
14. Type a label for the certificate, such as `ibmwebspheremqnyqm`, as shown in Figure 7-18.


Enter a label for the certificate:

ibmwebspheremqnyqm

Figure 7-18 The label for the certificate

The certificate now appears in the Signer Certificates repository, as shown in Figure 7-19 on page 178.


IBM Key Management - [C:\IBM\SSL\key.jks]

Key Database File Create View Help

Key database information

DB-Type: JKS database file

File Name: C:\IBM\SSL\key.jks

Token Label:

Key database content

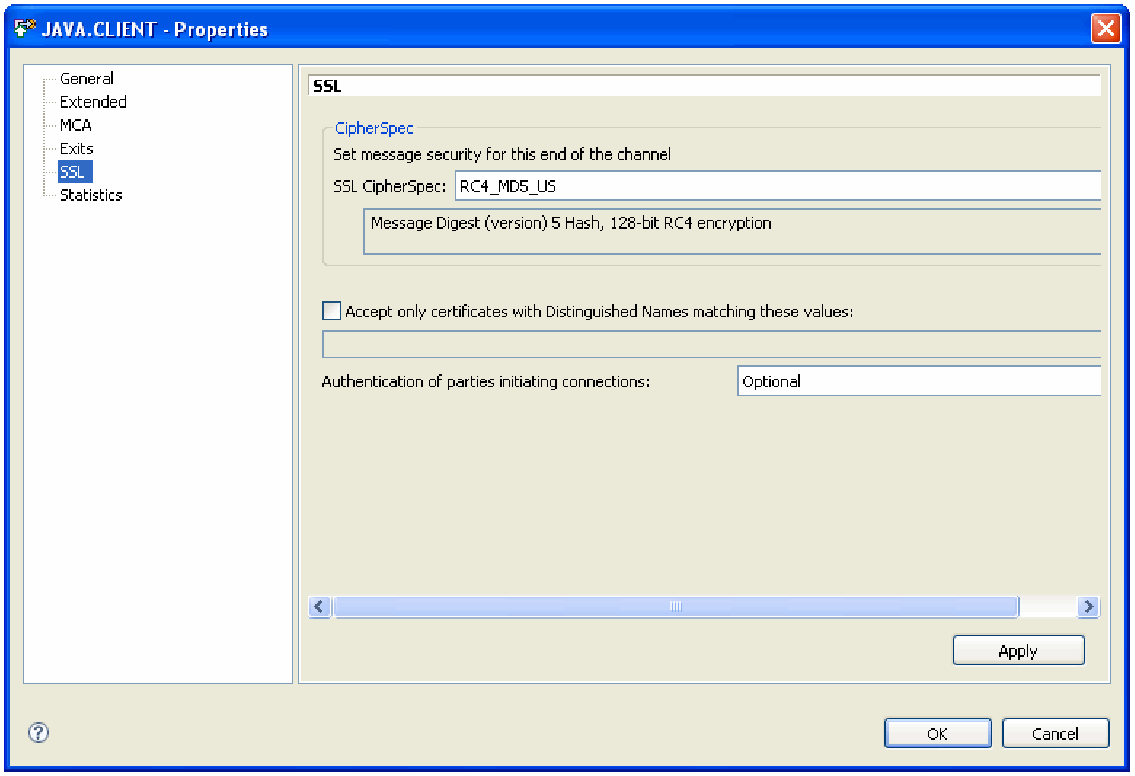
Signer Certificates

verisign class 3 public primary certification authority
verisign class 3 public primary certification authority - g3
verisign class 3 public primary certification authority - g2
thawte premium server ca
verisign class 1 public primary certification authority
entrust.net global secure server certification authority
thawte personal basic ca
thawte personal premium ca
thawte personal freemail ca
verisign class 2 onsite individual ca
ibmwebspheremqnyqm
verisign international server ca - class 3
thawte server ca
entrust.net certification authority (2048)
entrust.net client certification authority
entrust.net secure server certification authority

A signer certificate is from a certification authority (CA) or from another web site.

Figure 7-19 Self-signed certificate in the Signer Certificates List

15. Select one server-connection channel for the NYFIN client agent with SSL. Set the channel SSL CipherSpec, for example, choose **RC4_MD5_US**, as shown in Figure 7-20 on page 179. Click **OK**.



SSL

CipherSpec

Set message security for this end of the channel

SSL CipherSpec: RC4_MD5_US

Message Digest (version) 5 Hash, 128-bit RC4 encryption

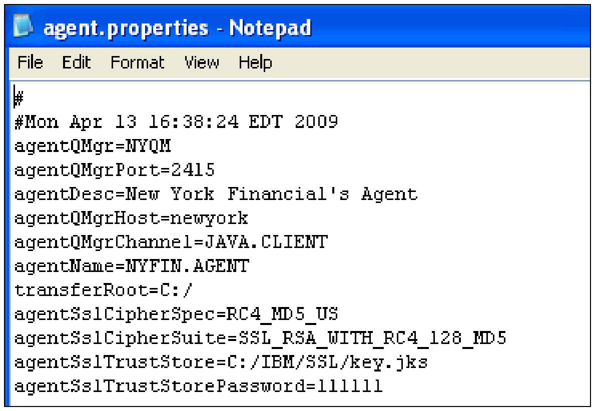
☐ Accept only certificates with Distinguished Names matching these values:

Authentication of parties initiating connections: Optional

Figure 7-20 Server-connection channel SSL properties

16. Add SSL properties to the agent.properties file on the NYFIN machine. Open the agent.properties file in the directory <FTE configuration data>\config\WASHQM\agents\NYFIN.AGENT, edit the contents, as shown in Figure 7-21 on page 180, and save the file.

Note: During an FTE install the agent's default configuration is defined and the FTE installer references it in the <FTE_install>/install.properties file. This default configuration is expressed as `<FTE configuration data>`.



```
#
#Mon Apr 13 16:38:24 EDT 2009
agentQMgr=NYQM
agentQMgrPort=2415
agentDesc=New York Financial's Agent
agentQMgrHost=newyork
agentQMgrChannel=JAVA.CLIENT
agentName=NYFIN.AGENT
transferRoot=C:/
agentSslCipherSpec=RC4_MD5_US
agentSslCipherSuite=SSL_RSA_WITH_RC4_128_MD5
agentSslTrustStore=C:/IBM/SSL/key.jks
agentSslTrustStorePassword=111111
```

Figure 7-21 Agent.properties of FTE client agent on NYFIN machine

17. Start NYFIN.AGENT, and check the output log for results.

After JKHL enables SSL security on the NY_FIN agent, JKHL starts secure file transfers to receive ad hoc reports of changes to office system passwords.

7.5 Multi-hop file transfers

Multi-hop transfers allow transfers to be sent from one point to another using an intermediary location.

7.5.1 The requirement

JKHL has some financial report files transferred from the Mumbai office to the New York Financial office, but the files cannot be directly sent between two offices. All files must be transferred to the New York financial office by the New York office.

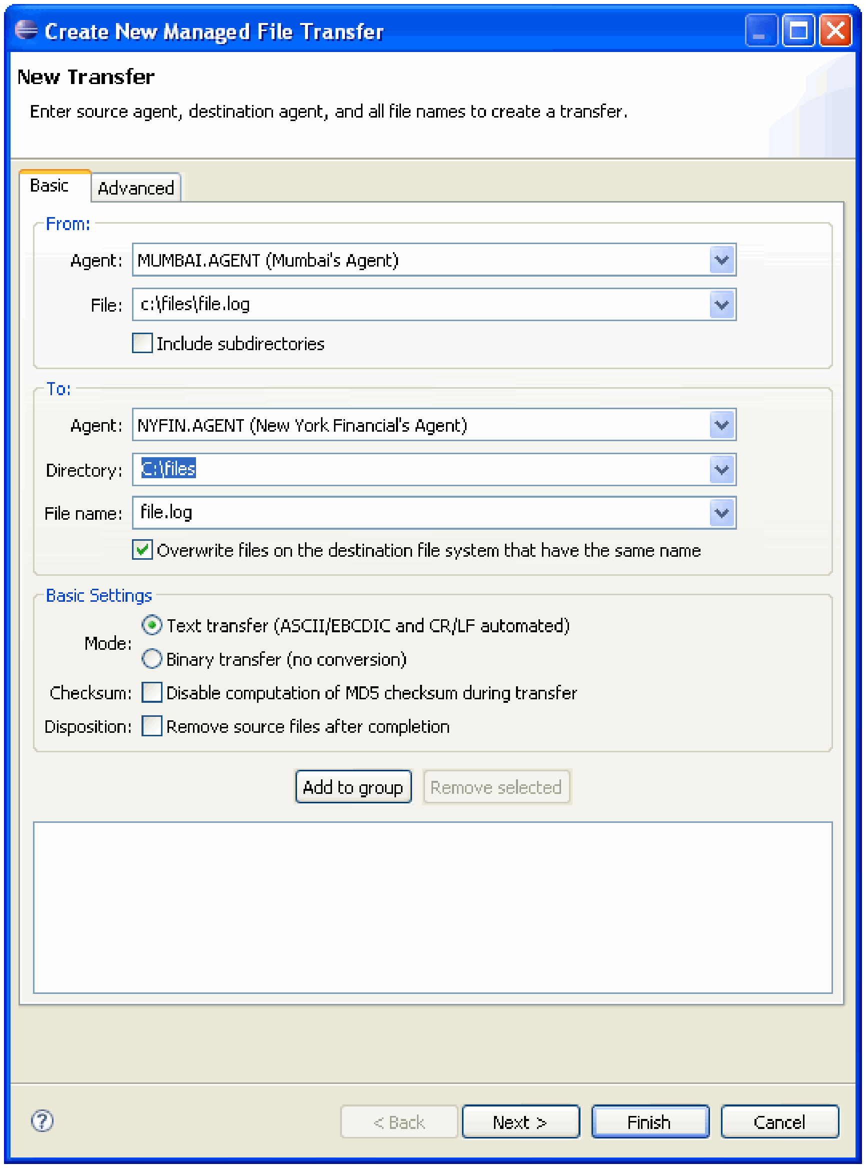
7.5.2 The solution

FTE helps JKHL implement the route from Mumbai to NY_FIN. At the same time, every office can transfer files by multi-hop to destinations. Multi-hop depends on WebSphere MQ's routing.

7.5.3 Testing file transfers from the Mumbai to NewYork Financial office

Before sending the financial report files, we first used the **`ftePingAgent`** command to check if the target agent is currently running and can respond. Follow the steps for the scenario:

1. On the Mumbai machine, open the command line, and switch the current directory to `bin` in the FTE installation directory.
2. On the Mumbai machine, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click it to launch the Eclipse GUI.
3. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **WASHQM**.
4. To launch the wizard, right-click **Transfer Templates**, and select **New Transfer**.
5. In the Creating New File Transfer wizard:
   a. Select the source agent name **MUMBAI.AGENT** in the agent list.
   b. Specify the full path to the text file on the source agent.
   c. Select the destination agent name **NYFIN.AGENT** in the agent list.
   d. Specify a receiving directory name on the destination agent.
   e. Specify a receiving file name.
   f. Enable the **Overwrite files on the destination file system that have the same name** option.
   g. In the Basic Settings option, choose **Text transfer** mode.
   h. Click **Finish**, and start the file transfer, as shown in Figure 7-22 on page 182.



Create New Managed File Transfer

New Transfer

Enter source agent, destination agent, and all file names to create a transfer.

Basic | Advanced

From:

Agent: MUMBAI.AGENT (Mumbai's Agent)

File: c:\files\file.log

[ ] Include subdirectories

To:

Agent: NYFIN.AGENT (New York Financial's Agent)

Directory: C:\files

File name: file.log

[✓] Overwrite files on the destination file system that have the same name

Basic Settings

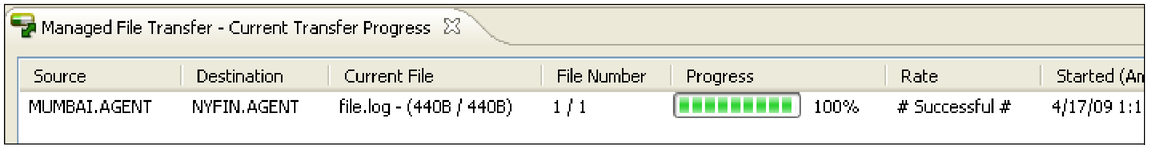
Mode: [● Text transfer (ASCII/EBCDIC and CR/LF automated)] ○ Binary transfer (no conversion)

Checksum: [ ] Disable computation of MD5 checksum during transfer

Disposition: [ ] Remove source files after completion

Figure 7-22 Multi-hop file transfer request

6. In the Current Transfer Progress view, display the success transfer from Mumbai to NYFIN, as shown in Figure 7-23 on page 183.


Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress	Rate	Started (An
MUMBAI.AGENT	NYFIN.AGENT	file.log - (440B / 440B)	1 / 1	100%	# Successful #	4/17/09 1:1

Figure 7-23   The result of multi-hop transfer from Mumbai to NYFIN

Now the scenario is complete and successful. JKHL can send business files from Mumbai to the New York financial office by multi-hop.

7.6 Setting WebSphere MQ channel compression

You can use WebSphere MQ File Transfer Edition to compress large files, which reduces the burden on network bandwidth.

7.6.1 The requirement

JKHL will transfer some large business files using text format. The sizes of these files are normally above 10 MB. If JKHL's network is busy, the large files are processed slowly.

7.6.2 The solution

Compressing an MQ channel can accomplish the requirement to reduce the amount of network traffic and therefore improve the file transfer performance on FTE.

The compression techniques that WebSphere MQ can provide are:

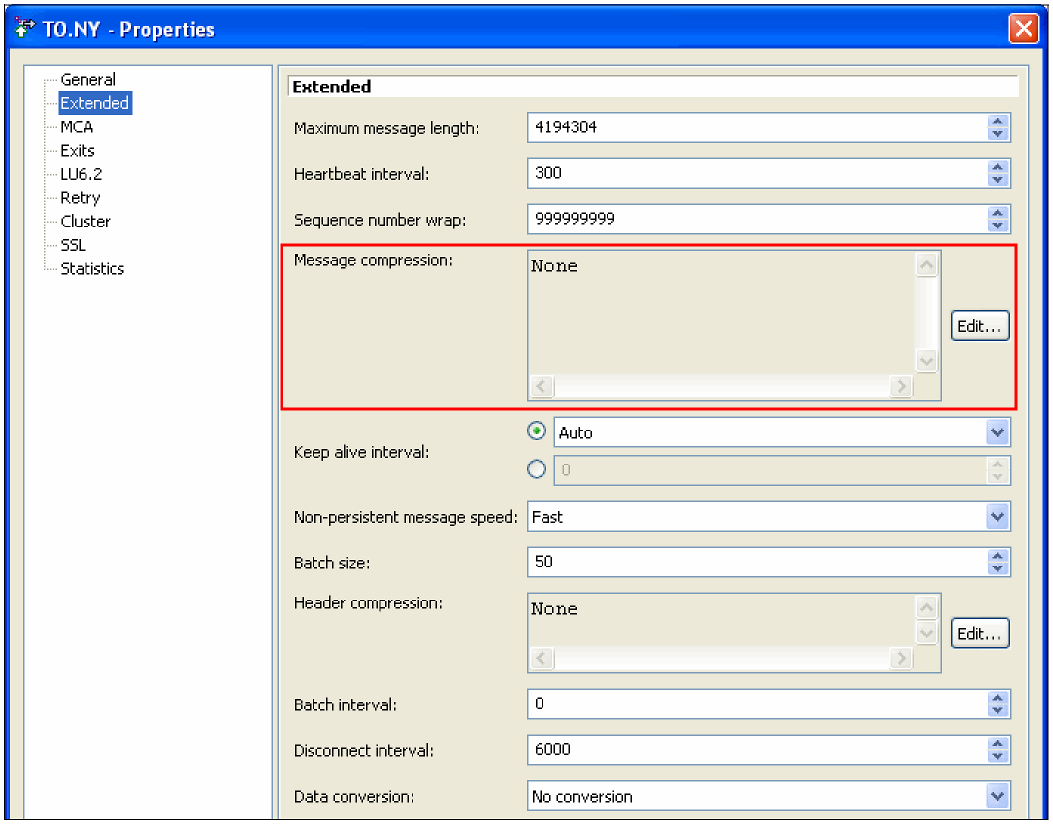
- RLE: Message data compression is performed using run-length encoding.
- ZLIBFAST: Message data compression is performed using the zlib compression technique. A fast compression time is preferred.
- ZLIBHIGH: Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

7.6.3 Configuring WebSphere MQ compression

For saving network bandwidth, we suggest that WebSphere MQ channel compression can be used in this scenario. The 1 GB large financial business document is transferred from Mumbai to New York.

To set WebSphere MQ channel compression:

1. On the Mumbai machine, open WebSphere MQ Explorer, and then click **IBM WebSphere MQ** → **Queue Managers** → **MUMBAIQM** → **Advanced** → **Channels**.
2. Right-click the cluster sender channel TO.NY on MUMBAIQM, and select **Properties**.
3. Choose the Extended view and review the message compression property, as shown in Figure 7-24.



Extended

Maximum message length: 4194304

Heartbeat interval: 300

Sequence number wrap: 999999999

Message compression: None

Keep alive interval: (●) Auto ( ) 0

Non-persistent message speed: Fast

Batch size: 50

Header compression: None

Batch interval: 0

Disconnect interval: 6000

Data conversion: No conversion

Figure 7-24 Channel Extended view

4. On the message compression property, click **Edit**, as shown in Figure 7-25 on page 185. Choose the ZLIBFAST compression technique of four items: None, RLE, ZLIBFAST, and ZLIBHIGH. Click **OK**.

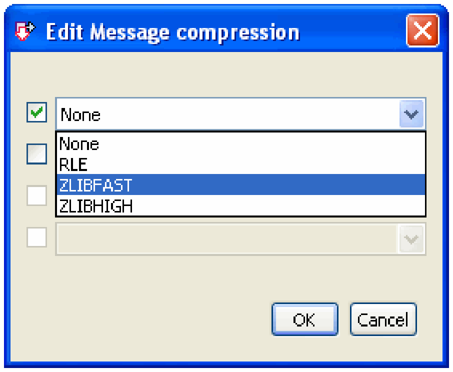
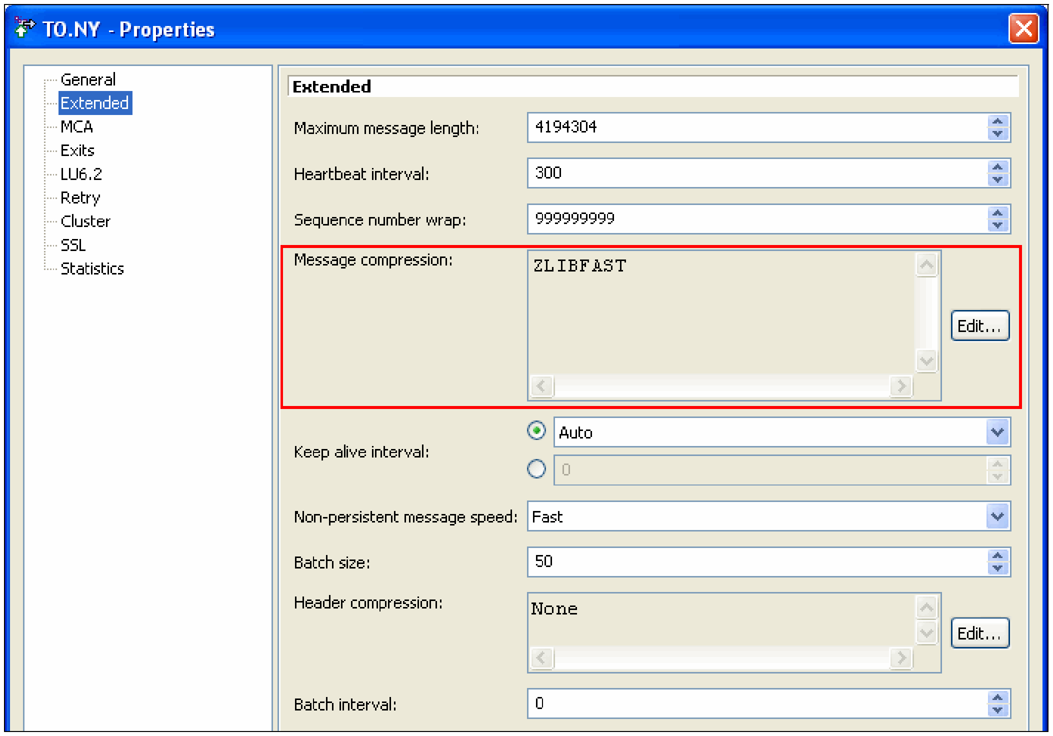


Figure 7-25 Compression selection

5. Finish the compression settings and view the current property, as shown in Figure 7-26.



Extended

Maximum message length: 4194304

Heartbeat interval: 300

Sequence number wrap: 999999999

Message compression: ZLIBFAST

Keep alive interval: Auto

Non-persistent message speed: Fast

Batch size: 50

Header compression: None

Batch interval: 0

Figure 7-26 The selection from message compression property

6. Change the message compression property on the receiver channel TO.NY of NYQM queue manager. Restart the cluster sender and receiver channels.

Now JKHL can start 1 GB large business file transfers from Mumbai to New York with MQ channel compression.

7.7 Checkpoint restart

Checkpoint restart allows stalled file transfers to restart from where they left off.

7.7.1 The requirement

The network is not stable between the JKHL offices. JKHL wants to ensure financial report file integrity when the network is broken and then recovered again.

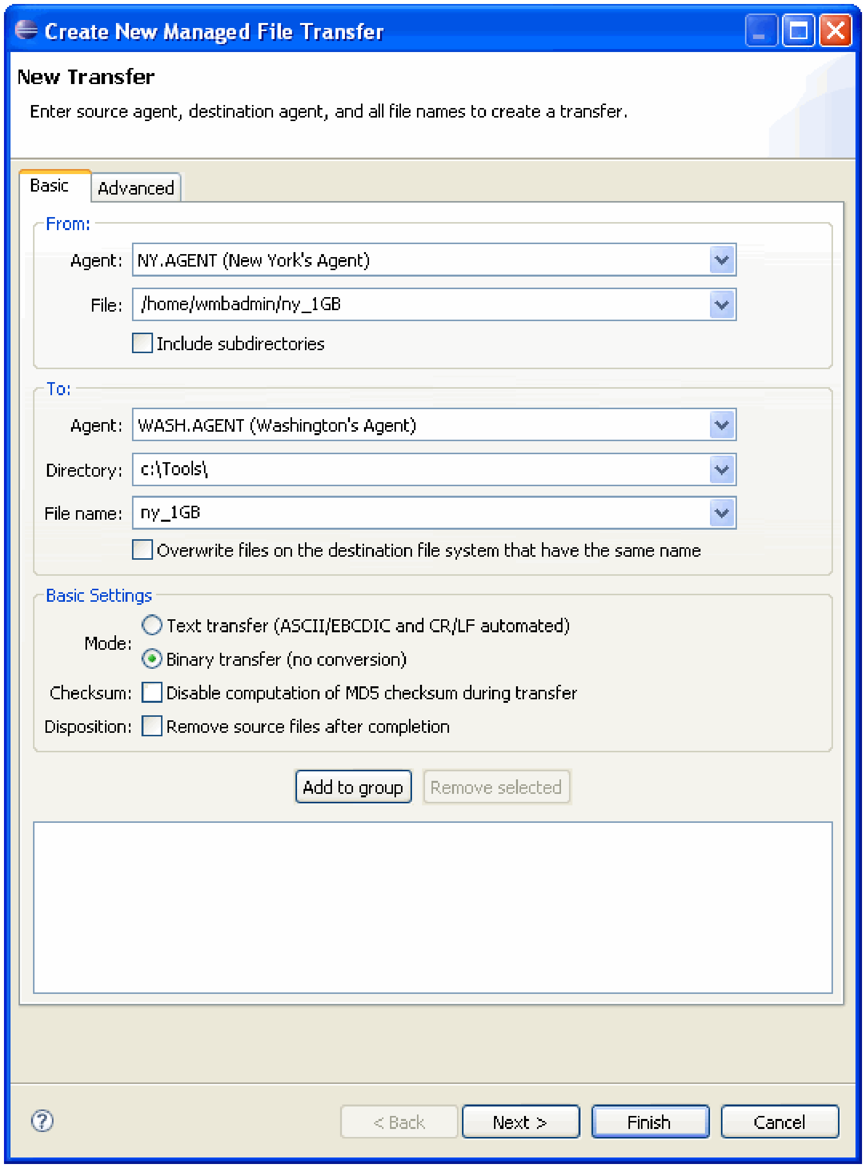
7.7.2 The solution

FTE can satisfy this requirement. FTE takes periodic check points during a transfer and saves the current file in the transfer and position within the file. When a transfer is restarted, the point at which it is restarted is determined using the check points and by re-synchronizing with the other agent that is participating in the transfer.

7.7.3 Testing checkpoint restart on FTE agents

In the following steps, JKHL begins to test the checkpoint mechanism of WebSphere MQ File Transfer Edition:

1. On the Mumbai machine, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **WASHQM**.
3. Right-click **Transfer Templates**, and select **New Transfer** to launch the wizard.
4. In the Creating New File Transfer wizard, shown in Figure 7-27 on page 187:
   a. Select the source agent name **NY.AGENT** in the agent list.
   b. Specify 1 GB file path on the source agent.
   c. Select the destination agent name **MUMBAI.AGENT** in the agent list.
   d. Specify a receiving directory name on the destination agent.



Create New Managed File Transfer

New Transfer

Enter source agent, destination agent, and all file names to create a transfer.

Basic | Advanced

From:

Agent: NY.AGENT (New York's Agent)

File: /home/wmbadmin/ny_1GB

[ ] Include subdirectories

To:

Agent: WASH.AGENT (Washington's Agent)

Directory: c:\Tools\

File name: ny_1GB

[ ] Overwrite files on the destination file system that have the same name

Basic Settings

Mode: ( ) Text transfer (ASCII/EBCDIC and CR/LF automated) [✓ Binary transfer (no conversion)]

Checksum: [ ] Disable computation of MD5 checksum during transfer

Disposition: [ ] Remove source files after completion

Figure 7-27 Starting a large file transfer to test FTE checkpoint

e. Specify a receiving file name.

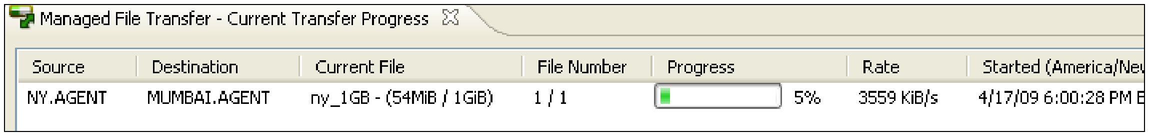
f. For the Basic Settings option, choose **Binary transfer** mode.

g. Enable the **Overwrite files on the destination file system that have the same name** option.

h. Clear the **Disable computation of MD5 checksum during transfer** option.

i. Click **Finish**, and start the file transfer.

5. See the **Current Transfer Progress** view, and display the state, as shown in Figure 7-28.


Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress	Rate	Started (America/Ne
NY.AGENT	MUMBAI.AGENT	ny_1GB - (54MiB / 1GiB)	1 / 1	5%	3559 KiB/s	4/17/09 6:00:28 PM E

Figure 7-28 The current transfer state

6. On Windows, click **Start** → **Settings** → **Network Connections**, and disable the **Local Area Connection** option for several minutes, as shown in Figure 7-29.

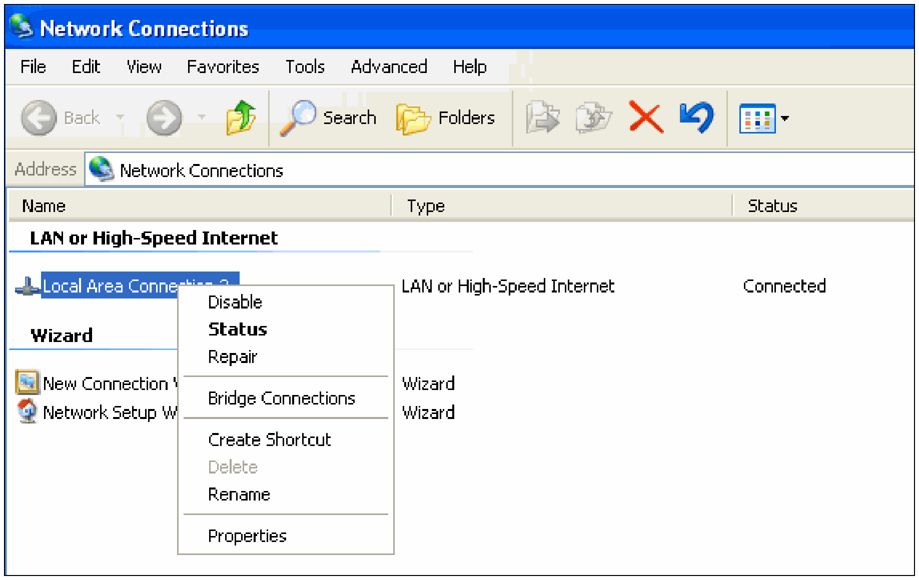
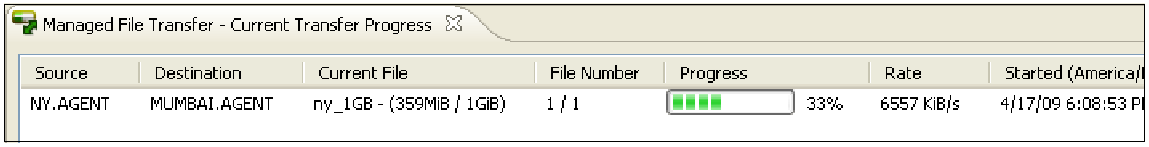


Figure 7-29 Disabling the local network on Windows

7. Enable the local network after several minutes, and wait for a while before the MQ channels are running again.

   When the channels recover between Mumbai and New York, the file transfer goes on, as shown in Figure 7-30 on page 189.


Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress	Rate	Started (America/I
NY.AGENT	MUMBAI.AGENT	ny_1GB - (359MiB / 1GiB)	1 / 1	33%	6557 KiB/s	4/17/09 6:08:53 PI

Figure 7-30 Recovering File transfer

Now the recovery is complete and successful. JKHL is very satisfied with the mechanism of WebSphere MQ File Transfer Edition.

7.8 Summary

JKHL built a basic file transfer system on WebSphere MQ File Transfer Edition. The requirements that FTE satisfied are:

- Basic secure and reliable file transfers
- File transfers using priorities
- Enabling SSL security on FTE client agents
- Multi-hop file transfers
- Enabling channel compression for file transfers
- File transfer recovery

8

Phase 2: Multi-step transfers

In this chapter, we describe a file transfer scenario that includes multiple steps and uses WebSphere MQ File Transfer Edition's *resource monitor* feature to automate those steps. This chapter is based on the JKHL Travel company and its four offices. As with other scenario chapters in this book, this chapter discusses a scenario that leverages an existing WebSphere MQ network so that WebSphere MQ File Transfer Edition can be used to automate existing file transfer systems and make them more reliable and audit-able.

In the next chapter, the JKHL company adds a fifth office, in Tokyo, that will also be integrated into its FTE network.

The topics that we discuss in this chapter are:

The solutions that we describe in this chapter illustrate how you can use the resource monitor feature of WebSphere MQ File Transfer Edition to integrate and automate file transfers with existing business systems.

8.1 Scenario overview

The fictional JKHL company has a number of branch offices. Each JKHL office does business in its local geography or region and each generates a *daily financial report* that is sent to the JKHL New York IT Headquarters for consolidation and processing, which we illustrate in Figure 8-1.

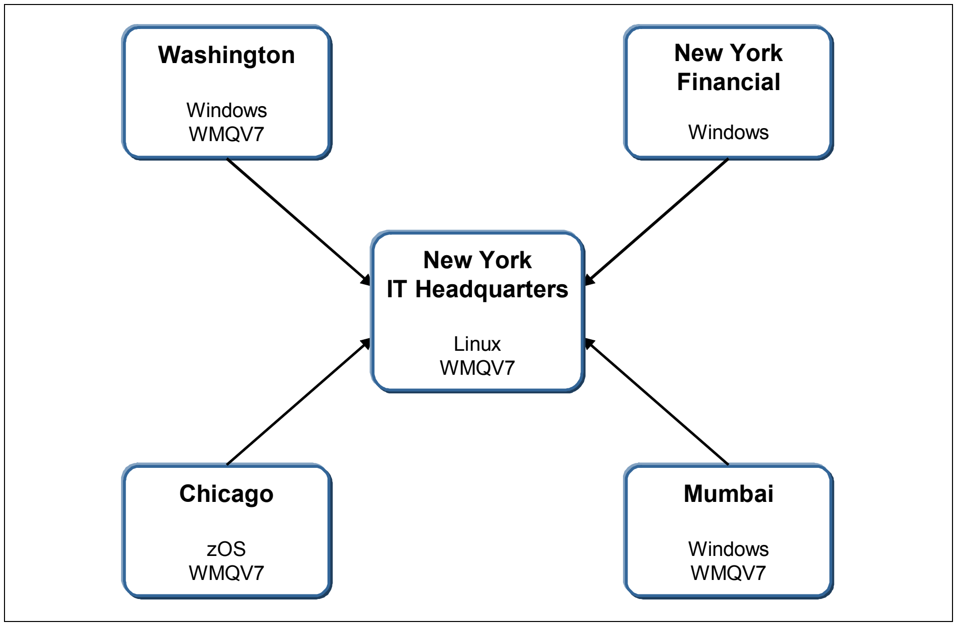


Figure 8-1 The JKHL company network

JKHL traditionally transferred these financial reports using a set of scripts that use simple FTP. We can summarize the existing file transfer process at JKHL as:

1. Each day, the JKHL branch offices generate a *daily financial report*.
2. After it is generated, the daily financial report is sent to New York IT Headquarters using FTP.
3. At New York IT Headquarters, the reports for all branch offices are processed by a *Consolidated Business Program* that generates a *daily consolidated data file*.

4. The daily consolidated data file is transferred to Chicago where the zOS mainframe processes can perform further processing.

8.1.1 Problems with JKHL's existing file transfer system

The JKHL IT staff is unhappy with the existing file transfer arrangements. They are aware of the following problems with it:

- The FTP scripts that send the daily reports to New York IT headquarters are difficult to maintain by JKHL's non-technical office staff.
- Each branch must manually run the scripts and each does so at different times. If a branch sends their file too late, the New York IT headquarters processing proceeds without it and generates an incomplete daily consolidated file.
- It is difficult for the non-technical JKHL office staff to know whether the FTP script ran successfully and that their file was delivered correctly. Unnoticed network errors often interrupt these transfers resulting in incomplete or incorrect processing at the New York IT headquarters.
- The distributed, multi-step nature of the process must be timed carefully to ensure that all files for the step are in place and complete before the next step can proceed. JKHL has difficulty ensuring that each step is correct and complete before the next step begins.

JKHL already has WebSphere MQ installed at some of its branch offices, and its IT staff wants to install WebSphere MQ File Transfer Edition to streamline and automate all aspects of the existing file transfer arrangements. We describe their proposal to overhaul JKHL's file transfer arrangements in the next sections.

8.2 Solution overview using WebSphere MQ File Transfer Edition

JKHL's IT staff designed a solution using WebSphere MQ File Transfer Edition that can replace the existing file transfer arrangements. They know that the New York IT headquarters and three of their branch offices (Washington, Mumbai, and Chicago) have WebSphere MQ installed and that these branches are using WebSphere MQ V6 or V7. With this in mind, they plan to use WebSphere MQ File Transfer Edition to replace the existing, manually executed FTP scripts with automated transfers.

Each office will run a WebSphere MQ File Transfer Edition *agent* that will manage all file transfer operations to and from the local office. To achieve this, they plan to install WebSphere MQ File Transfer Edition:

- For the offices that have a WebSphere MQ queue manager (for example, New York IT headquarters, Washington, Mumbai, and Chicago), they will install a WebSphere MQ File Transfer Edition *server* product.
- For the New York financial office, which does not have a WebSphere MQ queue manager, they plan to install a WebSphere MQ File Transfer Edition *client* product.

Figure 8-2 shows the JKHL WebSphere MQ File Transfer Edition Agents.

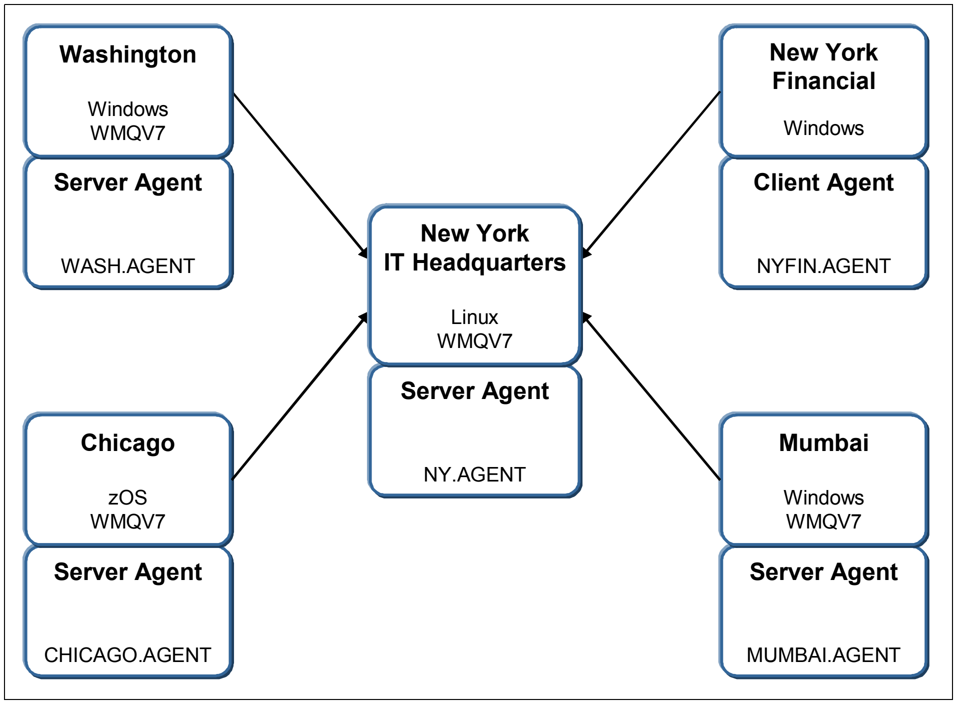


Figure 8-2 JKHL WebSphere MQ File Transfer Edition Agents

The proposed design has three steps:

- Daily Financial Report Collection

  In this step, we use FTE to transfer each office's daily financial report to New York IT headquarters.

- Trigger generation of the Daily Consolidated Data File

  In this step, we use FTE to trigger the JKHL Business Program that generates the Daily Consolidate Data file after all of the office financial reports are collected

- Daily Consolidated Data File Delivery to Chicago

  In this step, we use FTE to deliver the daily consolidated data file from New York IT headquarters to the mainframe system that is running at the Chicago office.

The proposed solution uses WebSphere MQ File Transfer Edition agents to leverage JKHL's existing WebSphere MQ infrastructure as much as possible. Because not all branch offices have WebSphere MQ channels running between them, the design also includes a plan to create those channels.

8.3 Preparing WebSphere MQ

WebSphere MQ File Transfer Edition uses WebSphere MQ to transport files between different systems. To allow WebSphere MQ File Transfer Edition to transfer files between all of JKHL's systems, the design must ensure that *MQ channels* are defined where they are needed.

Broadly, there are three ways to set up WebSphere MQ to allow communication between a group of systems:

- Use an MQ cluster for all systems

  If all of JKHL's systems are members of an MQ cluster, WebSphere MQ can manage the channels automatically. Using an MQ cluster results in simpler administration and configuration of the MQ network. The benefits of MQ clusters quickly increase as the number of systems grows.

- Do not use an MQ cluster

  When MQ clustering is not being used, sender and receiver channels must be defined possibly between each system, and every other system. For networks with a small number of systems, it is relatively simple to define and manage these channels manually. The complexity of managing these channels grows quickly as the number of systems increases.

- A combination of clustered and non-clustered systems

  It is possible to only include some systems in an MQ cluster. For systems that are cluster members, WebSphere MQ will manage the channels automatically. Systems that are not cluster members need manually defined and managed sender and receiver channels so that they can communicate with the other systems in the cluster.

Table 8-1 shows JKHL's queue managers and their proposed role in the WebSphere MQ File Transfer Edition network that will be installed.

Table 8-1 JKHL's Queue Managers

Office	Queue Manager	Platform	WMQ Version	Role
New York IT HQ	NYQM	Linux	V6	Agent
Washington	WASHQM	Windows	V7	Coordinator and Agent
Mumbai	MUMBAIQM	Windows	V7	Agent
Chicago	MQH1	zOS	V7	Agent
New York Financial		Windows		Agent *(client)*

The Washington office will host the designated WebSphere MQ File Transfer Edition *coordination queue manager.*

Looking at Table 8-1, New York IT headquarters, Washington, Mumbai, and Chicago will run an WebSphere MQ File Transfer Edition *server agent*, (because they have a local queue manager), while New York Financial will run an WebSphere MQ File Transfer Edition *client agent*.

8.3.1 Setting up channels

Setting up MQ channels to allow direct communication between all of JKHL's offices is required so that WebSphere MQ File Transfer Edition agents on any office system can transfer files to any other office system. Even with a small number of offices, however, this can become complex. For each office queue manager, sender channels to every other office queue manager must be defined and some arrangements made to start them when needed. In addition to creating sender channels, a matching set of receiver channels is also needed.

Figure 8-3 on page 197 shows JKHL's MQ network when it is not using clusters.

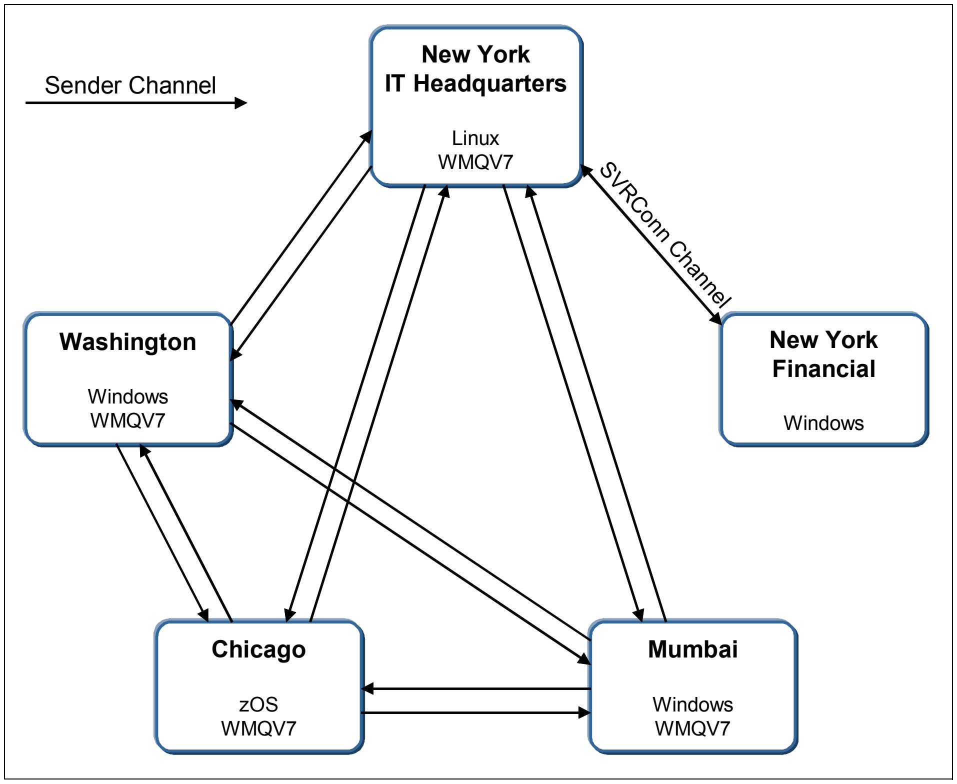


Figure 8-3 JKHL's MQ network when not using clusters

For JKHL's network, with four offices running servers, the IT staff must define twelve sender channels. Depending on how the MQ administrator chooses to design the network, perhaps four or more receiver channels must also be defined. The New York Financial office connects using a single server-connection (SVRCONN) to the New York IT headquarters.

Using MQ clusters, the number of channels that must be manually defined is greatly reduced to four sender channels and four receiver channels, plus the single New York Financial Server-connection channel. All other channel connections, such as those channels linking Mumbai to New York, are created and started automatically by the MQ cluster feature in each of the queue managers. See Figure 8-4 on page 198.

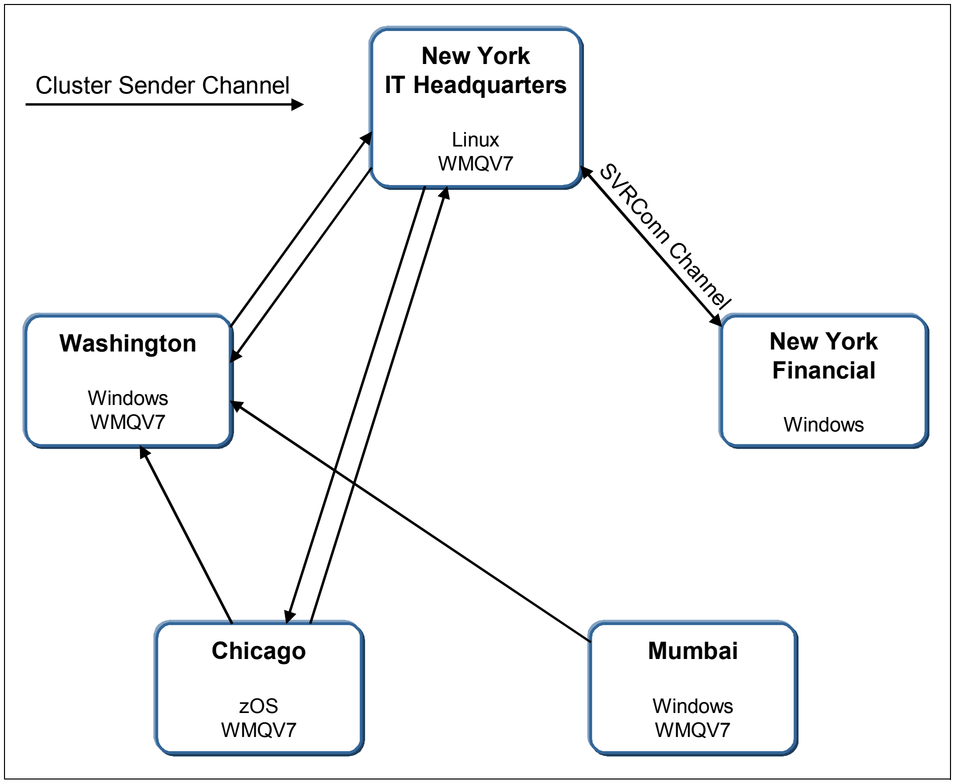


Figure 8-4 JKHL's network using MQ Clusters

For scenarios with larger numbers of office queue managers than JKHL, using MQ clusters offers even greater administrative savings.

8.4 Scenario setup

In this section, we outline how the JKHL office systems are configured for Phase 2 of this scenario. We briefly summarize what WebSphere MQ File Transfer Edition software is installed on each system and outline the WebSphere MQ queue managers that are involved at each location.

For guidance on performing this installation and configuration, refer to Chapter 3, "Installing and configuring WebSphere MQ File Transfer Edition" on page 33.

Installing and configuring the Washington Office

The Washington office has an existing WebSphere MQ Version 7 installation. Its queue manager is called `WASHQM`, and it is the designated coordination queue manager for the JKHL company FTE network.

For Washington, the following set up steps must be performed to set up FTE:

1. Install WebSphere MQ File Transfer Edition *Server*.

   Install the WebSphere MQ File Transfer Edition Server software, and during the installation, create the `WASH.AGENT`.

2. Install WebSphere MQ File Transfer Edition *Tools*, and configure MQ Explorer.

   Confirm that MQ Explorer is connecting to the `WASHQM` coordination queue manager correctly.

3. Configure WASHQM to be the coordinating queue manager.

   Use the MQSC script called WASHQM.mqsc in the WebSphere MQ File Transfer Edition *config* directory.

4. Start the Washington Agent using the **`fteStartAgent`** command.
5. Ping the Washington Agent using the **`ftePingAgent`** command.

Installing and configuring New York, Mumbai, and Chicago

1. Install WebSphere MQ File Transfer Edition *Server*.

   For New York, Mumbai, and Chicago install the WebSphere MQ File Transfer Edition *server* product.

2. For Mumbai, you can also install WebSphere MQ File Transfer Edition *Tools*, and configure MQ Explorer.

   Confirm that MQ Explorer is connecting to the `WASHQM` coordination queue manager correctly.

3. During installation (or later), create the Agent for each system.
4. Start the Agent.
5. Ping the Agent.

Installing and configuring the New York Financial office

1. Install WebSphere MQ File Transfer Edition *Client*.

   Install the WebSphere MQ File Transfer Edition Client software, and during the installation create the NYFIN.AGENT.

2. Optionally, install WebSphere MQ File Transfer Edition *Tools*, and configure MQ Explorer if it is available.

If MQ Explorer is installed, confirm that it is connecting to the `WASHQM` coordination queue manager correctly.

3. During installation (or later), create the New York Financial Agent called NYFIN.AGENT.
4. Start the New York Financial Agent.
5. Ping the New York Financial Agent.

8.5 Daily financial report collection

Figure 8-5 shows JKHL's offices with their WebSphere MQ File Transfer Edition agents and queue managers. We now use WebSphere MQ File Transfer Edition to set up automated file transfers that will send JKHL's daily financial reports to the New York IT headquarters.

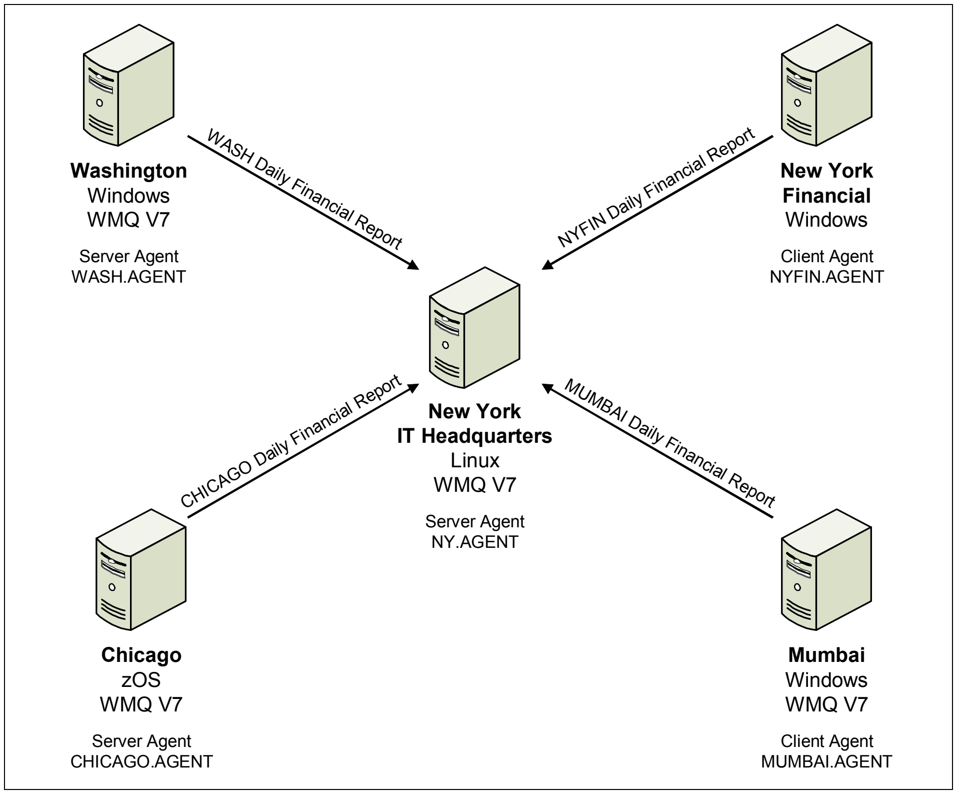


Figure 8-5 Daily Financial Report Collection solution using WebSphere MQ File Transfer Edition

For each branch office's agent, JKHL will define a **`resource monitor`** that watches for the presence of a daily financial report file. An WebSphere MQ File Transfer Edition resource monitor can be created using either command-line commands or the MQ Explorer interface.

Figure 8-6 shows the solution to the Daily Financial Report Collection step with file system paths at each participating node.

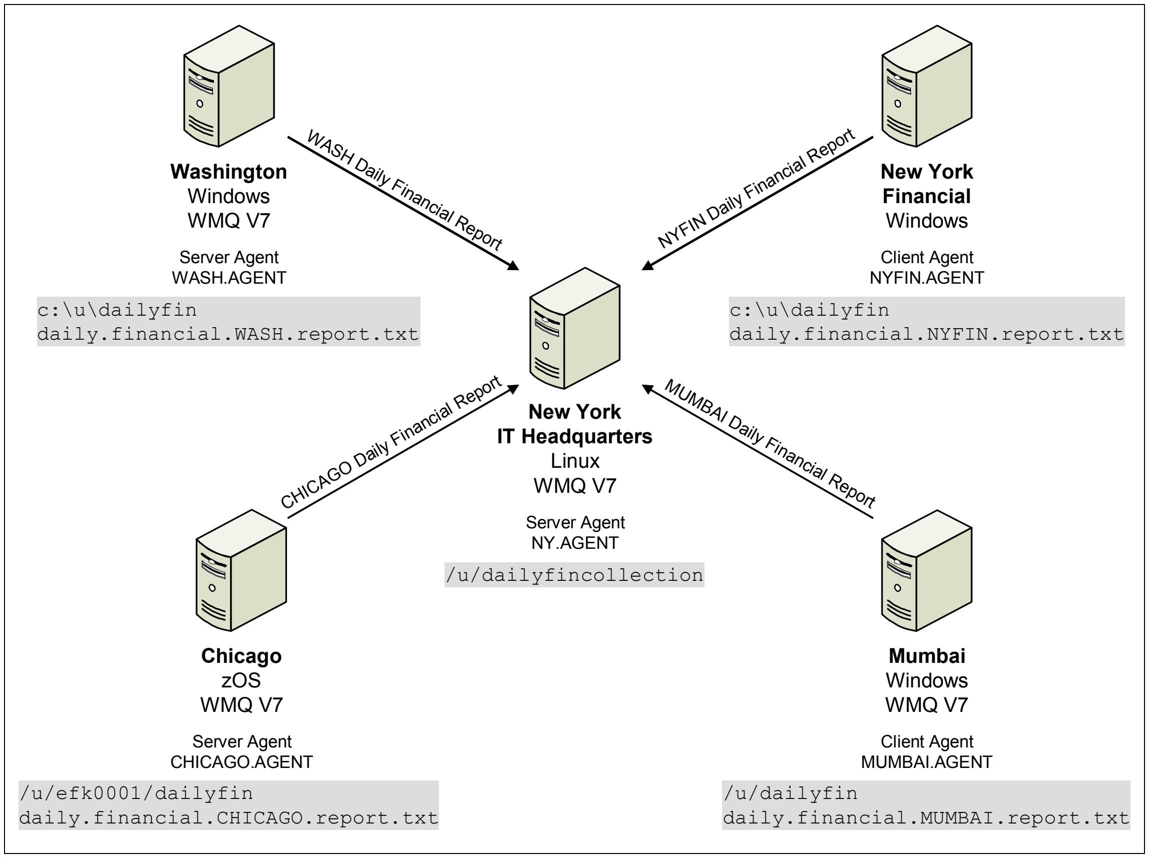


Figure 8-6 Daily Financial Report Collection solution showing file system paths

8.5.1 Setting up the Washington office using the command-line

In this section, we set up a resource monitor to automatically send Washington's daily financial report to New York. The transfer is automatically triggered when the analyst places the report file in the correct directory. To achieve this we:

- Create the required Transfer Specification XML
- Create the required resource monitor using the Transfer Specification

Creating the Washington office's Transfer Specification

The *Transfer Specification* is a definition of what file or files must be transferred, and what options apply to those transfers. The Transfer Specification is an XML file that you can manually create using an editor, or you can create it using the WebSphere MQ File Transfer Edition **fteCreateTransfer** command with the -gt switch.

For this scenario, we must transfer Washington's daily financial report from c:\u\dailyfin\daily.financial.WASH.report.txt to the New York IT headquarter's office system into location /u/dailfincollection. Because the daily financial report files contain text, the transfer specification must request *code-page conversion*, which performs any character encoding conversion that is needed, but it also converts the file's end-of-line characters.

1. To use the **fteCreateTransfer** command to create this transfer specification, at the Washington machine, enter the command in Example 8-1.

Example 8-1 Using fteCreateTransfer to create Washington's Transfer Specification

```
fteCreateTransfer
      -sa WASH.AGENT
      -sm WASHQM
      -da NY.AGENT
      -dm NYQM
      -df /u/dailyfincollection/daily.financial.WASH.report.txt
      -t text
      -de overwrite
      -sd leave
      -gt WASH.TS.xml
      c:\u\dailyfin\daily.financial.WASH.report.txt
```

   This command creates the transfer XML in a file called WASH.TS.xml in the current directory.

2. The command in Example 8-1 generates the transfer specification in WASH.TS.xml, which we show in Example 8-2.

Example 8-2 Washington's transfer specification generated by fteCreateTransfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
         <hostName>DIRAC.raleigh.ibm.com.</hostName>
         <userID>david</userID>
```

```
</originator>
    <sourceAgent agent="WASH.AGENT"
                 QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT"
                      QMgr="NYQM"/>
    <transferSet>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\daily.financial.WASH.report.txt</file>
        </source>
        <destination type="file" exist="overwrite">
         <file>/u/dailyfincollection/
                     daily.financial.WASH.report.txt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Of course, you can also manually create the Transfer Specification using an editor and the WebSphere MQ File Transfer Edition XML schema.

Creating the Washington office's resource monitor

In the previous section, we created a Transfer Specification XML file that describes the file names and paths to be transferred, the destination agent and queue manager, and the source agent and queue manager.

In this section, we describe how to create a WebSphere MQ File Transfer Edition *resource monitor* that performs the transfer when the monitor is triggered.

The **`fteCreateMonitor`** command is used to create a resource monitor on a given Agent. Recall that a WebSphere MQ File Transfer Edition resource monitor monitors a specified directory and executes its transfer if its *match-conditions* are met.

To prevent the transfer of a partial daily financial report, we use a *trigger file*. The trigger file is called daily.financial.WASH.report.txt.trigger, and its presence in the directory tells the WebSphere MQ File Transfer Edition Agent that the financial report arrived and the file transfer can begin. The trigger file's contents do not affect the operation of the transfer. The trigger file is created by the Washington office *after* the daily financial report is created in the correct c:\u\dailyfin directory.

Summarizing what the Washington resource monitor does, as shown in Figure 8-7:

1. The Washington agent runs the resource monitor.
2. The resource monitor monitors a single directory called:

   c:\u\dailyfin
3. In this directory, the monitor looks for files that meet its *match condition* which, in this case, is the presence of a single trigger file.
4. When the monitor's match conditions are satisfied, the monitor executes the transfer(s).
5. The monitor checks for match conditions periodically using a specified time interval.

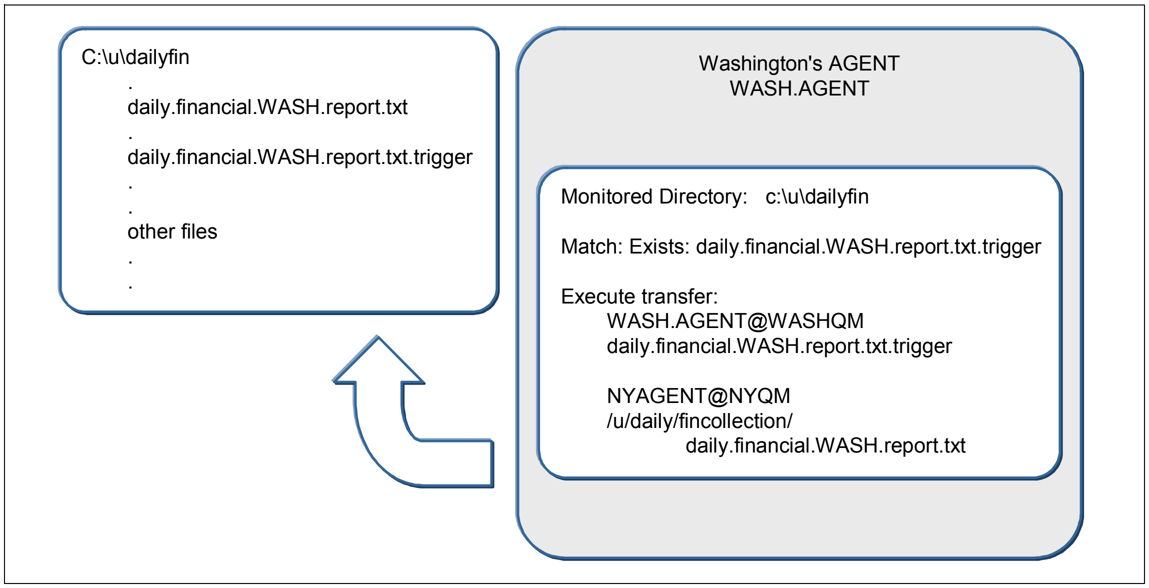

C:\u\dailyfin
.
daily.financial.WASH.report.txt
.
daily.financial.WASH.report.txt.trigger
.
.
other files
.
.
Washington's AGENT
WASH.AGENT
Monitored Directory: c:\u\dailyfin
Match: Exists: daily.financial.WASH.report.txt.trigger
Execute transfer:
WASH.AGENT@WASHQM
daily.financial.WASH.report.txt.trigger
NYAGENT@NYQM
/u/daily/fincollection/
daily.financial.WASH.report.txt

Figure 8-7 Washington's resource monitor for Daily Financial Report Collection

In this case, we use a resource monitor to execute the transfer of the daily financial report only if the trigger file exists:

1. Enter the commands in Example 8-3. This **fteCreateMonitor** command creates Washington's resource monitor.

Example 8-3 Using the fteCreateMonitor command to create Washington's resource monitor

```
fteCreateMonitor
     -ma WASH.AGENT
```

```
    -mm WASHQM
    -md c:\u\dailyfin
    -mn WASH_DailyFin_Monitor
    -mt WASH.TS.xml
    -pi 1
    -pu minutes
    -tr match,daily.financial.WASH.report.txt.trigger
```

When the command in Example 8-3 on page 204 is run, Washington's Agent begins to monitor the c:\u\dailyfin directory, looking for the specified trigger file. The monitor scans the directory once every minute. Because initially, the trigger file (called daily.financial.WASH.report.txt.trigger) does not yet exist, no transfer begins.

2. To test the Washington's monitor, create a report file with the correct name in c:\u\dailyfin on Washington, and then create a trigger file (called daily.financial.WASH.report.txt.trigger) in the same directory.

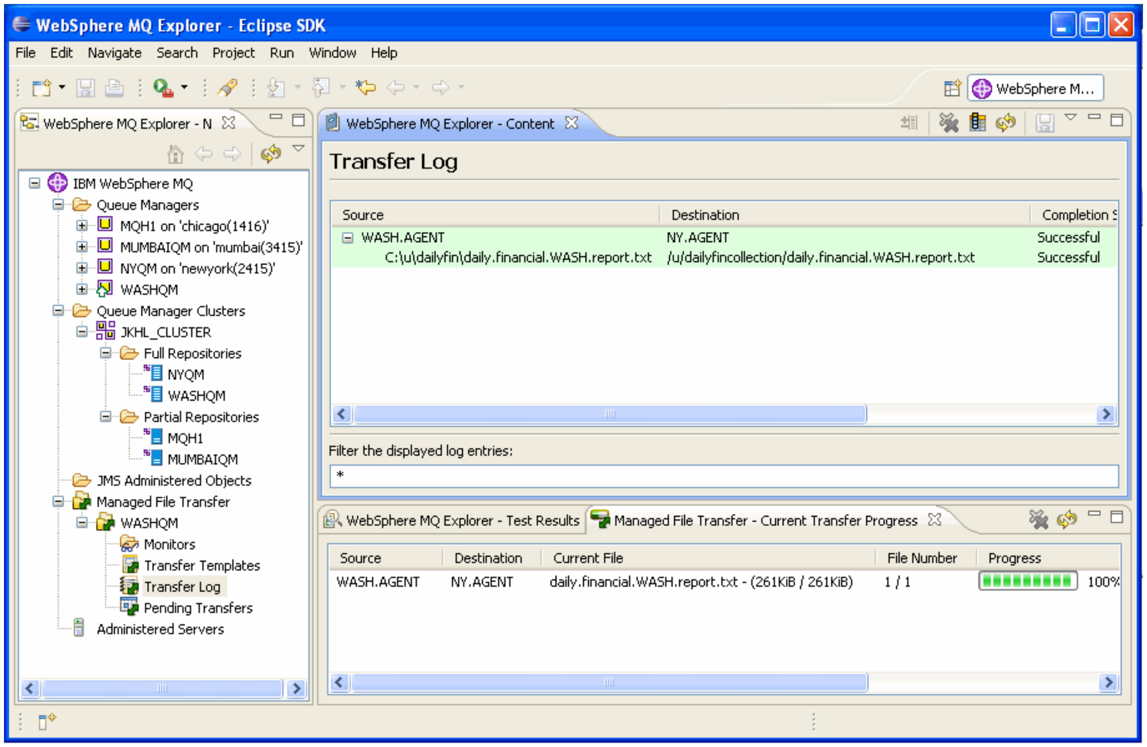
   Because Washington is using a Windows system, the easiest way to create the trigger files is to run the commands in Example 8-4 at a Windows console window.

Example 8-4 Windows commands to create a dummy trigger file

```
cd \u\dailyfin
echo “do it” > daily.financial.WASH.report.txt.trigger
```

   Within one minute, Washington's Agent (called WASH.AGENT) notices the existence of the trigger file and begins to execute the transfer.

3. You can see the transfer starting and running using MQ Explorer. Figure 3 shows the view from MQ Explorer after the above resource monitor triggers the file transfer.



Transfer Log

Source	Destination	Completion S
WASH.AGENT	NY.AGENT	Successful
C:\u\dailyfin\daily.financial.WASH.report.txt	/u/dailyfincollection/daily.financial.WASH.report.txt	Successful

Filter the displayed log entries:
*

Source	Destination	Current File	File Number	Progress
WASH.AGENT	NY.AGENT	daily.financial.WASH.report.txt - (261KiB / 261KiB)	1 / 1	100%

Figure 8-8   MQ Explorer view after Washington's transfer is triggered

The Agent's resource monitor continues monitoring the c:\u\dailyfin directory but does not re-trigger the transfer unless the *trigger* file's *last modification date* or its *size* changes.

Transferring the trigger file

In this scenario, the daily financial reports from all offices are collected and processed by a business consolidation application (at New York IT headquarters). To allow the consolidation application to positively know that all of the files arrived and are complete, it is useful to also transfer the *trigger* file from each office to the new New York IT headquarter collection directory, which you can do by revising the Transfer Specification XML file, created in the section above called "Creating the Washington office's Transfer Specification" on page 202, to also transfer the trigger file *after* the daily report file is transferred. Basically, we must add a new item element in the WASH.TS.xml file.

To add a new item element in the WASH.TS.xml file:

1. Use the XML fragment in Example 8-5 to transfer the trigger file.

Example 8-5 XML fragment to transfer Washington's trigger file

```
<item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\
              daily.financial.WASH.report.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/dailyfincollection/
              daily.financial.WASH.report.txt.trigger</file>
        </destination>
</item>
```

   After you add the XML fragment to the WASH.TS.xml file, it will contain the source shown in Example 8-6.

   In Example 8-6, the bolded section of the XML shows how to request that the Washington office's trigger file also be transferred. Notice that the trigger file transfer is defined *after* the daily financial report is transferred.

Example 8-6 Improved Transfer Specification for Washington

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>DIRAC.raleigh.ibm.com.</hostName>
        <userID>david</userID>
    </originator>
    <sourceAgent agent="WASH.AGENT"
           QMgr="WASHQM"/>
    <destinationAgent agent="NY.AGENT"
                QMgr="NYQM"/>
    <transferSet>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\daily.financial.WASH.report.txt</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
                    daily.financial.WASH.report.txt</file>
        </destination>
      </item>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\
               daily.financial.WASH.report.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/dailyfincollection/
               daily.financial.WASH.report.txt.trigger</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

2. After revising the WASH.TS.xml file, delete the monitor and recreate it using the commands in Example 8-7.

Example 8-7 Re-creating the Washington monitor

```
fteDeleteMonitor WASH_DailyFin_Monitor

fteCreateMonitor
      -ma WASH.AGENT
      -mm WASHQM
      -md c:\u\dailyfin
      -mn WASH_DailyFin_Monitor
      -mt WASH.TS.xml
      -pi 1
      -pu minutes
      -tr match,daily.financial.WASH.report.txt.trigger
```

8.5.2 Setting up the Mumbai and Chicago offices using the command line

Setting up the Mumbai and Chicago systems to perform an automated file transfer is similar to the procedure that we used for the Washington system.

To set up the Mumbai and Chicago offices using the command line:

1. Create the Transfer Specification XML.
2. Create the resource monitor using the Transfer Specification.

Creating the Mumbai office's Transfer Specification

The simplest way to create the Transfer Specification for Mumbai is to use the Washington office's transfer specification (contained in WASH.TS.xml) in Figure 8-6 on page 201, as a template.

Alternatively, you can create Mumbai's Transfer Specification using a **`fteCreateTransfer`** command as we did for the Washington office.

1. Use the **`fteCreateTransfer`** command to create Mumbai's Transfer Specification, as shown in Example 8-8.

Example 8-8 Using fteCreateTransfer to create Mumbai's Transfer Specification

```
fteCreateTransfer
      -sa MUMBAI.AGENT
      -sm MUMBAIQM
      -da NY.AGENT
      -dm NYQM
      -df /u/dailyfincollection/daily.financial.MUMBAI.report.txt
      -t text
      -de overwrite
      -sd leave
      -gt MUMBAI.TS.xml
      c:\u\dailyfin\daily.financial.MUMBAI.report.txt
```

2. The command generates the Transfer Specification in MUMBAI.TS.xml. You must edit this file to include the transfer of Mumbai's trigger file. Example 8-9 shows MUMBAI's complete Transfer Specification.

Example 8-9 Transfer Specification for MUMBAI

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>DIRAC.raleigh.ibm.com.</hostName>
        <userID>david</userID>
    </originator>
    <sourceAgent agent="MUMBAI.AGENT"
                 QMgr="MUMBAIQM"/>
    <destinationAgent agent="NY.AGENT"
                      QMgr="NYQM"/>
    <transferSet>
      <item mode="text" checksumMethod="MD5">
```

```
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\daily.financial.MUMBAI.report.txt</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
                    daily.financial.MUMBAI.report.txt</file>
        </destination>
      </item>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\
               daily.financial.MUMBAI.report.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/dailyfincollection/
               daily.financial.MUMBAI.report.txt.trigger</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Creating the Mumbai office's resource monitor

The Mumbai office's resource monitor is modeled on the Washington monitor.

1. Create the resource monitor using the command in Example 8-10.

Example 8-10 Creating Mumbai's resource monitor

```
fteCreateMonitor
      -ma MUMBAI.AGENT
      -mm MUMBAIQM
      -md c:\u\dailyfin
      -mn MUMBAI_DailyFin_Monitor
      -mt MUMBAI.TS.xml
      -pi 1
      -pu minutes
      -tr match,daily.financial.MUMBAI.report.txt.trigger
```

When the command in Example 8-10 runs, Mumbai's Agent will begin monitoring the c:\u\dailyfin directory, looking for the defined trigger file. The monitor will scan the directory once every minute. Because initially, the trigger file (called daily.financial.MUMBAI.report.txt.trigger) does not yet exist, the transfer does not immediately begin.

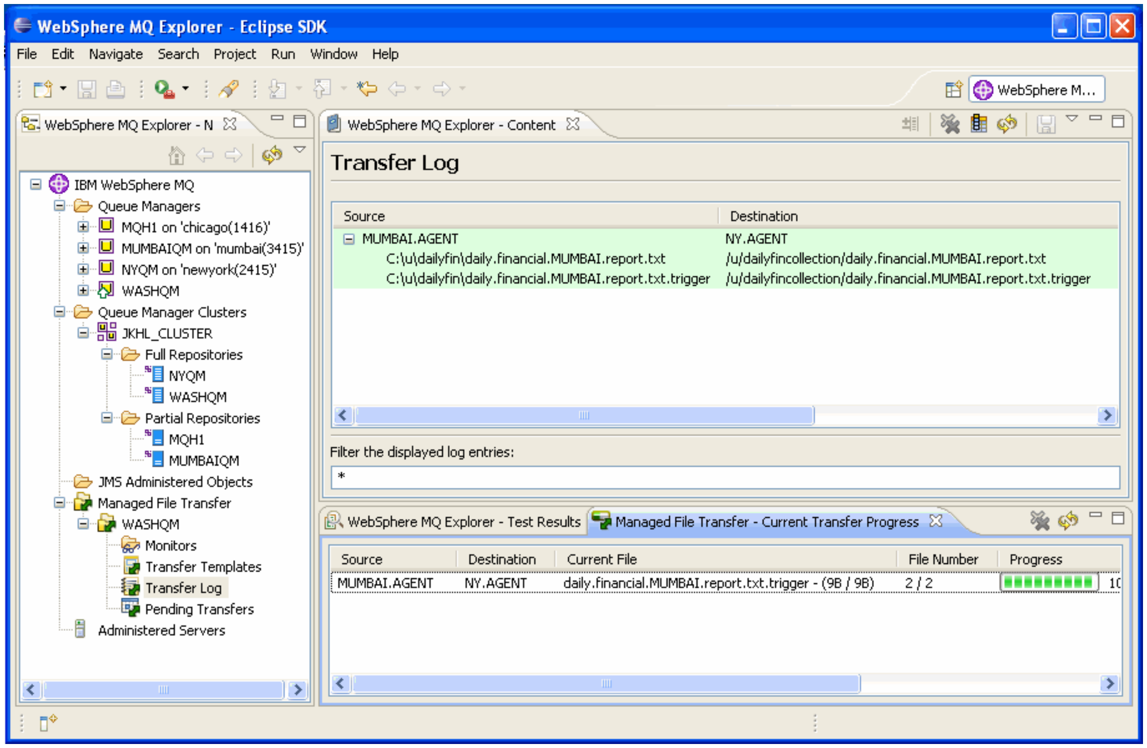
2. To test the monitor, create a report file with the correct name in c:\u\dailyfin, and then create a trigger file (called daily.financial.MUMBAI.report.txt.trigger) in the same directory.

   Since Mumbai is also using a Windows system, the easiest way to create the trigger is by running the commands in Example 8-11 at a Windows console window.

Example 8-11 Windows commands to create a dummy trigger file

```
cd \u\dailyfin
echo “do it” > daily.financial.MUMBAI.report.txt.trigger
```

3. Within one minute, Mumbai’s Agent (called MUMBAI.AGENT) will notice the existence of the trigger file and begin to execute the transfer. Figure 8-9 shows the MQ Explorer view of a triggered transfer from Mumbai to the New York IT headquarters.



Source	Destination
MUMBAI.AGENT	NY.AGENT
C:\u\dailyfin\daily.financial.MUMBAI.report.txt	/u/dailyfincollection/daily.financial.MUMBAI.report.txt
C:\u\dailyfin\daily.financial.MUMBAI.report.txt.trigger	/u/dailyfincollection/daily.financial.MUMBAI.report.txt.trigger

Source	Destination	Current File	File Number	Progress
MUMBAI.AGENT	NY.AGENT	daily.financial.MUMBAI.report.txt.trigger - (9B / 9B)	2 / 2	10

Figure 8-9 MQ Explorer view after the transfer is triggered

Creating the Chicago office's Transfer Specification

The simplest way to create the Transfer Specification for Chicago is to use the Transfer Specification from one of the other offices (for example Washington's Transfer Specification contained in WASH.TS.xml) as a template. Because Chicago is running on a z/OS UNIX System Services system, the directory paths on the destination side are UNIX format (forward slashes) rather than Windows format.

Alternatively, you can create Chicago's Transfer Specification using a **`fteCreateTransfer`** command as we did for Washington:

1. Use the **`fteCreateTransfer`** command to create Chicago's Transfer Specification, as shown in Example 8-12.

Example 8-12 Using fteCreateTransfer to create Chicago's Transfer Specification

```
fteCreateTransfer
      -sa CHICAGO.AGENT
      -sm MQH1
      -da NY.AGENT
      -dm NYQM
      -df /u/dailyfincollection/daily.financial.CHICAGO.report.txt
      -t text
      -de overwrite
      -sd leave
      -gt CHICAGO.TS.xml
      /u/efk0001/dailyfin/daily.financial.CHICAGO.report.txt
```

2. The command in Example 8-12 generates the following Transfer Specification in CHICAGO.TS.xml. You must edit this file to include the transfer of Chicago's trigger file. Example 8-9 on page 209 shows Chicago's completed transfer specification.

Example 8-13 Transfer Specification for Chicago

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>washington.raleigh.ibm.com.</hostName>
        <userID>david</userID>
    </originator>
    <sourceAgent agent="CHICAGO.AGENT"
                QMgr="MQH1"/>
    <destinationAgent agent="NY.AGENT"
                      QMgr="NYQM"/>
    <transferSet>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
           <file>/u/efk0001/
              dailyfin/daily.financial.CHICAGO.report.txt</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
                    daily.financial.CHICAGO.report.txt</file>
        </destination>
      </item>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
           <file>/u/efk0001/dailyfin/
                 daily.financial.CHICAGO.report.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
              daily.financial.CHICAGO.report.txt.trigger</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Creating the Chicago office's resource monitor

The Chicago office's resource monitor is modeled on the resource monitors that are defined for the other offices. To create the Chicago office's resource monitor:

1. Create the resource monitor using the command in Example 8-14.

Example 8-14 Creating Chicago's resource monitor

```
fteCreateMonitor
      -ma CHICAGO.AGENT
      -mm MQH1
      -md /u/efk0001/dailyfin
      -mn CHICAGO_DailyFin_Monitor
      -mt CHICAGO.TS.xml
      -pi 1
      -pu minutes
      -tr match,daily.financial.CHICAGO.report.txt.trigger
```

After the command in Example 8-14 on page 213 is run, Chicago's Agent will begin to monitor the /u/efk0001/dailyfin directory, looking for the defined trigger file. The monitor will scan the directory once every minute. Since initially, the trigger file (called daily.financial.CHICAGO.report.txt.trigger) does not yet exist, the transfer does not immediately begin.

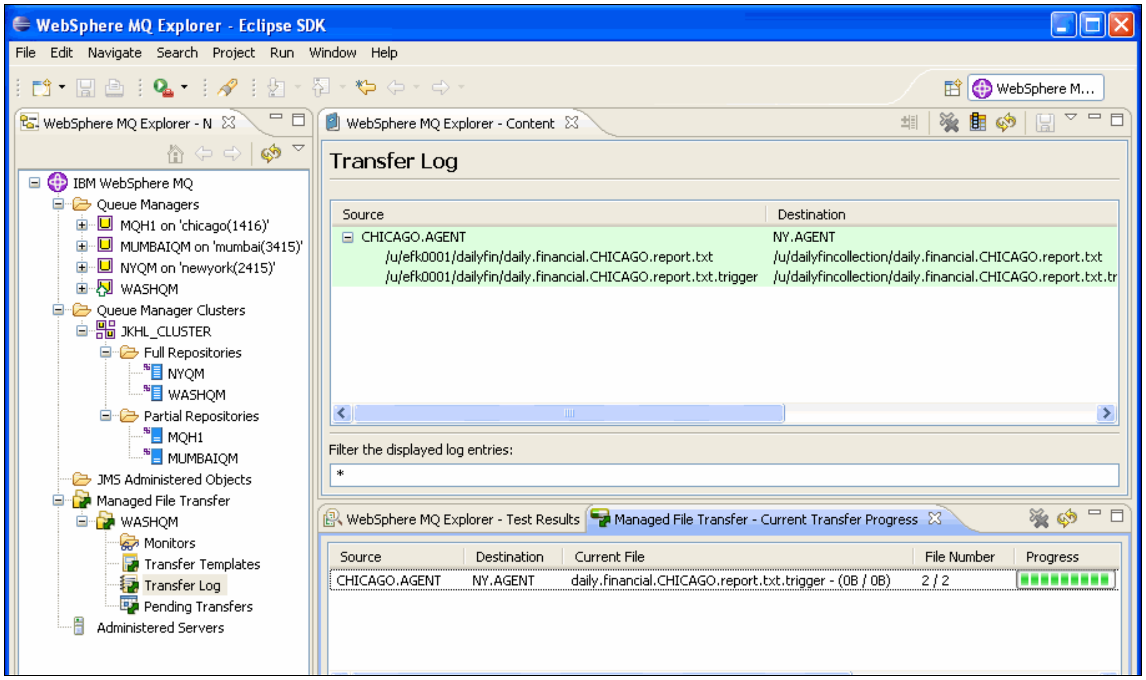
2. To test the monitor, create a report file with the correct name in /u/efk0001/dailyfin, and then create a trigger file (called daily.financial.MUMBAI.report.txt.trigger) in the same directory.

   Because Chicago is running on a z/OS UNIX System Services system, the easiest way to create the trigger is to run the commands in Example 8-15 at a z/OS shell prompt.

Example 8-15 zOS UNIX System Services system command to create a dummy trigger file

```
cd /u/efk0001/dailyfin
touch daily.financial.CHICAGO.report.txt.trigger
```

3. Within one minute, Chicago's Agent (called CHICAGO.AGENT) will notice the existence of the trigger file and begin to execute the transfer. Figure 8-10 shows the MQ Explorer view of a triggered transfer from Chicago to the New York IT headquarters.



Transfer Log

Source	Destination
CHICAGO.AGENT	NY.AGENT
/u/efk0001/dailyfin/daily.financial.CHICAGO.report.txt	/u/dailyfincollection/daily.financial.CHICAGO.report.txt
/u/efk0001/dailyfin/daily.financial.CHICAGO.report.txt.trigger	/u/dailyfincollection/daily.financial.CHICAGO.report.txt.tr

Filter the displayed log entries:
*

Source	Destination	Current File	File Number	Progress
CHICAGO.AGENT	NY.AGENT	daily.financial.CHICAGO.report.txt.trigger - (0B / 0B)	2 / 2	🟩

Figure 8-10 MQ Explorer view of a trigger transfer from Chicago

8.5.3 Setting up the New York Financial office using the command line

Setting up the New York Financial office system to perform an automated file transfer is slightly different from the other offices because New York Financial does not have a local queue manager; instead, the New York Financial office will use the New York IT headquarter's queue manager (NYQM) using a *client* connection.

In the examples in this section, you can see that the queue manager that we refer to is NYQM.

Creating New York Financial's Transfer Specification

As we described for the other offices, you can create the Transfer Specification using either an **fteCreateTransfer** command using the specification from one of the other office's as a template.

To create the New York Financial office's Transfer Specification:

1. Use the **fteCreateTransfer** command to create New York Financial's Transfer Specification, as shown in Example 8-16.

Example 8-16 Using fteCreateTransfer to create New York Financial's Transfer Specification

```
fteCreateTransfer
      -sa NYFIN.AGENT
      -sm NYQM
      -da NY.AGENT
      -dm NYQM
      -df /u/dailyfincollection/daily.financial.NYFIN.report.txt
      -t text
      -de overwrite
      -sd leave
      -gt NYFIN.TS.xml
      c:\u\dailyfin\daily.financial.NYFIN.report.txt
```

2. The command in Example 8-16 generates the Transfer Specification in Example 8-17, NYFIN.TS.xml. Edit this file to include the transfer of New York Financial's trigger file. Example 8-17 shows New York Financial's Transfer Specification.

Example 8-17 Transfer Specification for New York Financial

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>DIRAC.raleigh.ibm.com.</hostName>
         <userID>david</userID> </originator>
   <sourceAgent agent="NYFIN.AGENT"
               QMgr="NYQM"/>
   <destinationAgent agent="NY.AGENT"
                    QMgr="NYQM"/>
   <transferSet>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\u\dailyfin\daily.financial.NYFIN.report.txt</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
              daily.financial.NYFIN.report.txt</file>
        </destination>
      </item>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
           <file>c:\u\dailyfin\
              daily.financial.NYFIN.report.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
           <file>/u/dailyfincollection/
                 daily.financial.NYFIN.report.txt.trigger</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Creating New York Financial's resource monitor

The New York Financial office's resource monitor is modeled on the Washington office's monitor. To create New York Financial's resource monitor:

1. Create this resource monitor using the command in Example 8-18.

Example 8-18 Creating New York Financial's resource monitor

```
fteCreateMonitor
      -ma NYFIN.AGENT
      -mm NYQM
```

```
   -md c:\u\dailyfin
   -mn NYFIN_DailyFin_Monitor
   -mt NYFIN.TS.xml
   -pi 1
   -pu minutes
   -tr match,daily.financial.NYFIN.report.txt.trigger
```

When the command in Example 8-18 on page 216 is run, the New York Financial Agent begins to monitor the c:\u\dailyfin directory, looking for the defined trigger file. The monitor scans the directory once every minute. Because initially, the trigger file (called daily.financial.NYFIN.report.txt.trigger) does not yet exist, the transfer does not immediately begin.

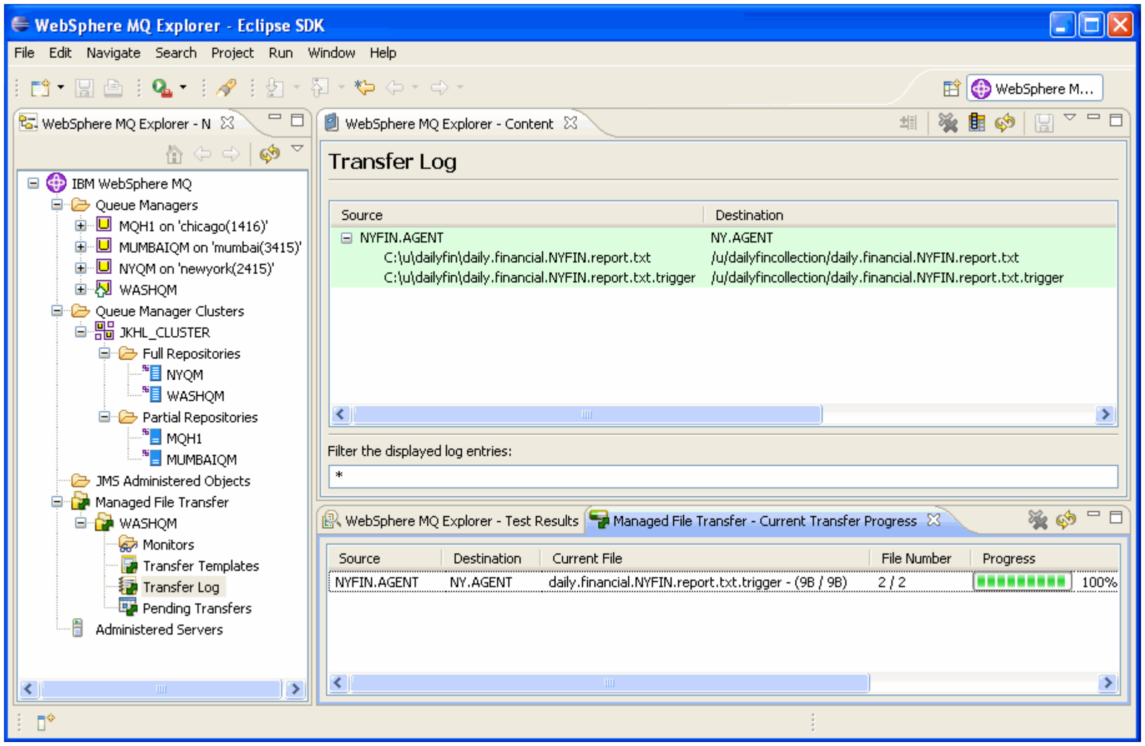
2. To test the monitor, create a report file with the correct name in c:\u\dailyfin, and then create a trigger file (called daily.financial.NYFIN.report.txt.trigger) in the same directory.

   Because New York Financial is using a Windows system, the easiest way to create the trigger is to run the commands in Example 8-19 at a Windows console window.

Example 8-19 Windows commands to create a dummy trigger file

```
cd \u\dailyfin
echo “do it” > daily.financial.NYFIN.report.txt.trigger
```

3. Within one minute, New York Financial’s Agent (called NYFIN.AGENT) will notice the existence of the trigger file and begin to execute the transfer. Figure 8-11 on page 218 shows the MQ Explorer view of a triggered transfer from the New York Financial office to the New York IT headquarters.



Source	Destination
NYFIN.AGENT	NY.AGENT
C:\u\dailyfin\daily.financial.NYFIN.report.txt	/u/dailyfincollection/daily.financial.NYFIN.report.txt
C:\u\dailyfin\daily.financial.NYFIN.report.txt.trigger	/u/dailyfincollection/daily.financial.NYFIN.report.txt.trigger

Source	Destination	Current File	File Number	Progress
NYFIN.AGENT	NY.AGENT	daily.financial.NYFIN.report.txt.trigger - (9B / 9B)	2 / 2	100%

Figure 8-11   MQ Explorer view of a trigger transfer from New York Financial

8.6 Trigger generation of the daily consolidated data file

In the second step of the scenario, we use FTE to trigger the JKHL Consolidated Business Program that generates the Daily Consolidated Data file. This trigger will occur after the daily financial reports for all offices are collected at New York IT headquarters.

To achieve this, we use a WebSphere MQ File Transfer Edition resource monitor to trigger the JKHL Consolidated Business Program when the last of the daily financial reports arrive at the New York IT HQ collection directory, as illustrated in Figure 8-12 on page 219.

In the JKHL company, the New York Financial office always prepares and sends their daily financial report after the reports from the other offices arrive. We exploit this fact when we design the FTE transfer definitions. In this section, we use the FTE resource monitor feature to automatically run the JKHL business program to consolidate the individual office financial reports.

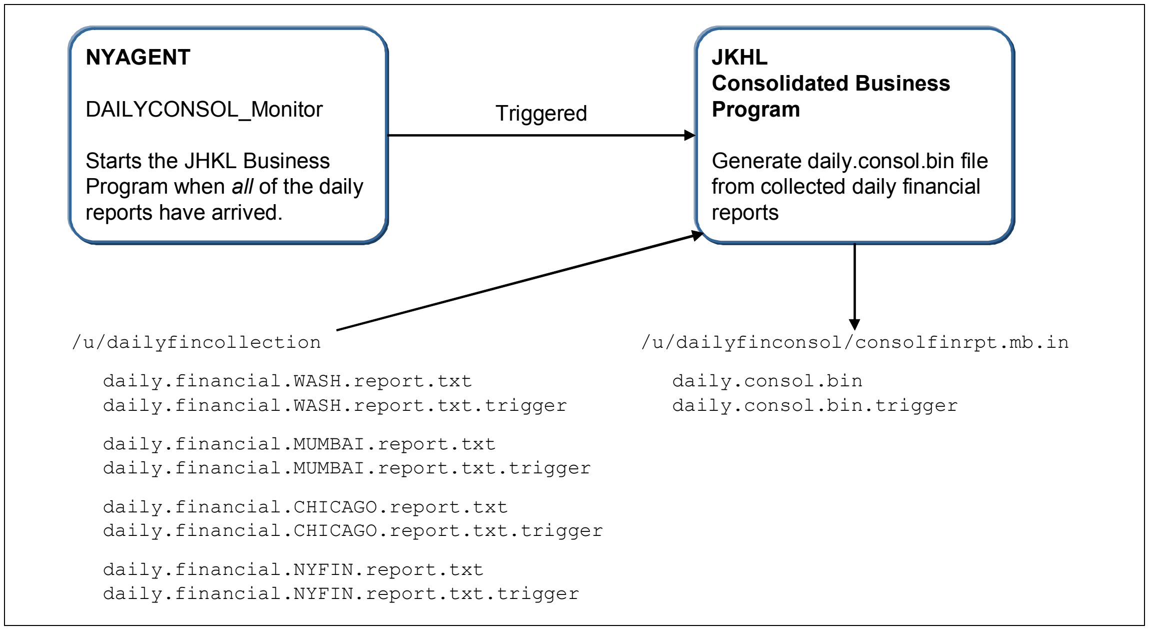

NYAGENT
DAILYCONSOL_Monitor
Starts the JHKL Business Program when *all* of the daily reports have arrived.
Triggered
JHKL Consolidated Business Program
Generate daily.consol.bin file from collected daily financial reports
/u/dailyfincollection
daily.financial.WASH.report.txt
daily.financial.WASH.report.txt.trigger
daily.financial.MUMBAI.report.txt
daily.financial.MUMBAI.report.txt.trigger
daily.financial.CHICAGO.report.txt
daily.financial.CHICAGO.report.txt.trigger
daily.financial.NYFIN.report.txt
daily.financial.NYFIN.report.txt.trigger
/u/dailyfinconsol/consolfinrpt.mb.in
daily.consol.bin
daily.consol.bin.trigger

Figure 8-12 Trigger the generation of the daily consolidated data file

8.6.1 Setting up the New York Agent using the command line

In this section, we discuss how to set up the New York agent to trigger the JKHL Consolidated Business Program when the New York Financial daily report becomes available.

To do this, we use WebSphere MQ File Transfer Edition's resource monitor feature to trigger a *command* when the NYFIN daily report trigger file is delivered. In this case, the command is a script called ConsolidatedBusinessProgram.sh that runs all of the JKHL company processing that is needed to perform the consolidation.

Configuring the Agent for managedCall

A *managedCall* is a WebSphere MQ File Transfer Edition transfer specification that calls a program, command, or Ant script, rather than initiating a file transfer.

The FTE agent can only execute programs and scripts that are located in a configured directory path. This path is defined by the commandPath attribute in the agent's agent.properties file. The ConsolidatedBusinessProgram.sh script that we reference in Example 8-20 on page 220, must be stored in a directory that the agent's commandPath attribute named.

Example 8-20 shows a dummy script that you can use to represent the JKHL ConsolidatedBusinessProgram.sh program. The script simply puts its output data in the daily.consol.bin file that is ready to be transferred in the next section. Additionally, the script updates a log file and, importantly, updates the trigger file that is needed to trigger the next transfer.

Example 8-20 The demonstration JKHL ConsolidatedBusinessProgram.sh script

```
#!/bin/sh
#
# Write some data to the daily.consol.bin file
# and append a line to log file for easy debugging
#

echo "$(date) ::: ConsolidatedBusinessProgram.sh running"
      | tee /u/dailyfinconsol/daily.consol.bin
      >> /u/dailyfinconsol/ConsolidatedBusinessProgram.log

# Touch the trigger file
touch /u/dailyfinconsol/daily.consol.bin.trigger

exit 0
```

To enable FTE to perform a managedCall on the above script in Example 8-20, store the ConsolidateBusinessProgram.sh script in the /u/dailyfinconsol directory headquarter's trigger file, as shown in Example 8-25 on page 224.

Creating the Start Business Program Transfer Specification

Using a managedCall transfer specification, you can call a program or shell script or run an Ant script. When a managedCall transfer specification is driven by a resource monitor, the invocation of the program or script can be triggered in the same flexible way that a file transfer can be triggered.

Currently managedCall transfer specifications can only be set up by creating the XML content manually, and in this section we show how to construct that XML.

To set up the managedCall transfer specification:

1. Use the transfer specification in Example 8-21 to start the JKHL Consolidated Business Program.

Example 8-21 NY_RunCBP.xml transfer specification

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>DIRAC@raleigh.ibm.com</hostName>
      <userID>david</userID>
    </originator>
    <agent QMgr="NYQM" agent="NY.AGENT"/>
    <reply QMGR="NYQM">reply</reply>
    <transferSet priority="1">
      <call>
      <command name="ConsolidatedBusinessProgram.sh"
         type="executable"
         retryCount="2" retryWait="30" successRC="0">
      </command>
      </call>
    </transferSet>
  </managedCall>
</request>
```

2. Store the XML in Example 8-21 on page 220 in a file called NY_RunCBP.xml. We use this file in the next section to create the resource monitor.

Creating the resource monitor to trigger the Consolidated Business Program

The **fteCreateMonitor** command creates the resource monitor that triggers the execution of the Consolidated Business Program:

1. Create the resource monitor using the command in Example 8-22. Notice that the trigger conditions specify that the New York Financial office daily financial report (trigger file) must be present in the New York IT headquarter's collection directory.

Example 8-22 Creating the resource monitor to enable triggering of the Consolidated Business Program.

```
fteCreateMonitor
      -ma NY.AGENT
      -mm NYQM
      -md /u/dailyfincollection
      -mn NY_Trigger_CBP_Monitor
      -mt NY.RunCBP.xml
      -pi 1
      -pu minute
      -tr match,daily.financial.NYFIN.report.txt.trigger
```

Examining the command in Example 8-22 on page 221, the -tr option refers to the New York Financial office's daily report *trigger file*. Using the -tr switch in this way ensures that the NYFIN daily financial report is present before running the program.

When the command in Example 8-22 on page 221 is run, the New York Agent begins to monitor the /u/dailyfincollection directory, looking for the particular trigger file. The monitor scans the directory once every minute. Because the trigger file does not initially exist, the program does not immediately start.

Scanning the directory: Real business scenarios do not scan the directory every minute. Because these files are typically only created and sent once each day, scanning the directory every hour or every two hours is sufficient.

For testing and evaluation, a short interval of one minute is more illustrative.

2. To test the monitor, provide the daily financial report files on each office system, and let the transfers that we defined in the previous section move the reports to the New York collection directory along with their associated trigger files.

   Ensure that the NYFIN daily report is delivered last.

   Alternatively, you can create dummy files in the New York collection directory, as shown in Example 8-23.

Example 8-23 Linux commands to create dummy report and trigger files

```
cd /u/dailyfincollection

echo “Dummy report data” > daily.financial.WASH.report.txt
echo “Dummy report data” > daily.financial.MUMBAI.report.txt
echo “Dummy report data” > daily.financial.CHICAGO.report.txt
echo “Dummy report data” > daily.financial.NYFIN.report.txt

touch daily.financial.NYFIN.report.txt.trigger
```

Within one minute, New York's Agent (called NY.AGENT) will notice the existence of the daily.financial.NYFIN.report.txt.trigger file and will begin executing the Consolidated Business Program.

8.7 Daily consolidated data file delivery

The third step of this scenario requires that the *daily consolidated data* file be delivered to Chicago from the New York IT headquarter system where it was generated. This file is generated by the business consolidation application that was described in the previous section.

The file transfer design for this step is simpler than the Daily Financial Report Collection step because it involves only a single file transfer (and its trigger file) between the New York IT headquarters and Chicago offices.

For this step, a single daily consolidated data file must be transferred from New York IT headquarter's directory /u/dailyfinconsol to Chicago's directory /u/efk0001/consolfinrpt.mb.in.

Figure 8-13 shows the Daily Consolidated Data file transfer with the relevant file system directory paths visible.

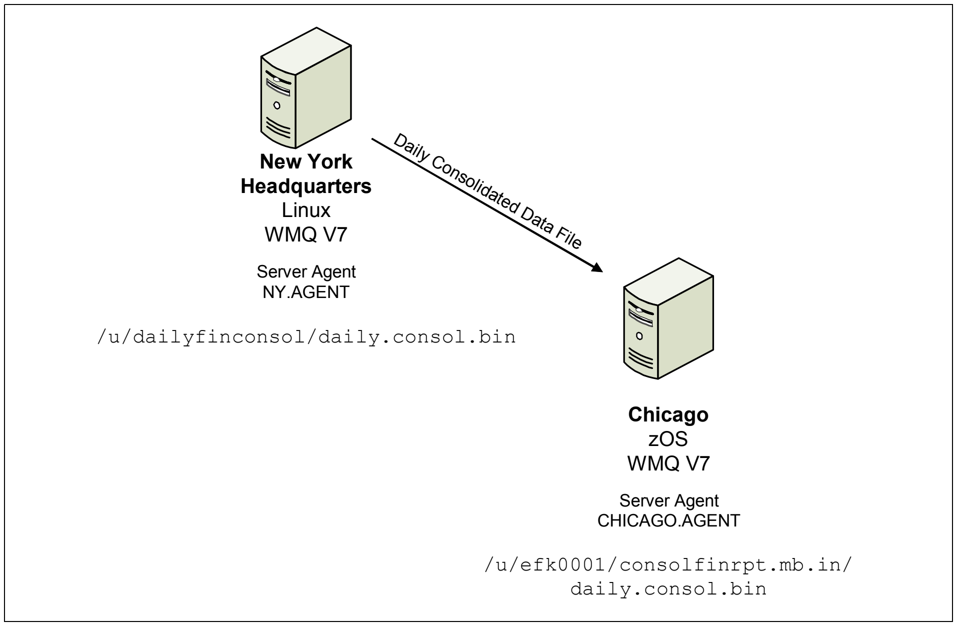

New York Headquarters
Linux
WMQ V7
Server Agent
NY.AGENT
/u/dailyfinconsol/daily.consol.bin
Daily Consolidated Data File
Chicago
zOS
WMQ V7
Server Agent
CHICAGO.AGENT
/u/efk0001/consolfinrpt.mb.in/daily.consol.bin

Figure 8-13 Daily Consolidated Data File file transfer with file system directory paths

8.7.1 Setting up New York IT headquarter Agent for the Daily Consolidated Data File Transfer

The steps to set up the New York IT headquarter Agent to perform the Daily Consolidated Data File Delivery file transfer are:

1. Create the Transfer Specification XML.
2. Create the resource monitor using the Transfer Specification.

Creating New York IT headquarter's Transfer Specification

The simplest way to create the Transfer Specification for the New York IT headquarter is to use the Washington office's Transfer Specification (contained in WASH.TS.xml) as a template, as shown in Example 8-6 on page 207.

Alternatively, you can create New York IT headquarter's Transfer Specification using a **`fteCreateTransfer`** command:

1. Use the **`fteCreateTransfer`** command to create New York IT headquarter's transfer specification, as shown in Example 8-24.

Example 8-24 Using fteCreateTransfer to create New York IT HQ's Daily Consolidated Transfer Specification

```
fteCreateTransfer
      -sa NY.AGENT
      -sm NYQM
      -da CHICAGO.AGENT
      -dm MQH1
      -df /u/efk0001/consolfinrpt.mb.in/daily.consol.bin
      -t binary
      -de overwrite
      -sd leave
      -gt DAILYCON.TS.xml
      /u/dailyfinconsol/daily.consol.bin
```

2. The command in Example 8-24 generates the following transfer specification in DAILYCON.TS.xml. Edit this file to include the transfer of New York IT headquarter's trigger file.

   Example 8-25 shows the complete Step 2 transfer Specification to move the daily consolidated data file from New York IT HQ to Chicago.

Example 8-25 The DAILYCON.TS.xml file

```
<request version="2.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
```

```
  <managedTransfer>
    <originator>
        <hostName>192.168.65.132</hostName>
        <userID>wmbadmin</userID>
    </originator>
    <sourceAgent agent="NY.AGENT"
                 QMgr="NYQM"/>
    <destinationAgent agent="CHICAGO.AGENT"
                 QMgr="MQH1"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/u/dailyfinconsol/daily.consol.bin</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/efk0001/consolfinrpt.mb.in/daily.consol.bin</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Creating New York IT headquarter's resource monitor

1. Enter the command shown in Example 8-26 to create the New York IT headquarter resource monitor for Daily Consolidated Data File Delivery.

Example 8-26 Creating the New York IT HQ resource monitor

```
fteCreateMonitor
      -ma NY.AGENT
      -mm NYQM
      -md /u/dailyfinconsol
      -mn DailyConsol_Monitor
      -mt DAILYCON.TS.xml
      -pi 1
      -pu minutes
      -tr match,daily.consol.bin.trigger
```

2. To test the monitor, create a consolidated data file with the correct name in /u/dailyfinconsol, and then create a trigger file (called daily.consol.bin.trigger) in the same directory.
3. Because New York IT headquarter is running a Linux system, the easiest way to create the trigger is to run the commands in Example 8-27 on page 226 at a shell prompt.

Example 8-27 Linux commands to create a dummy trigger file

```
cd /u/dailyfinconsol
touch daily.consol.bin.trigger
```

Within one minute, new York HQ's Agent (called NY.AGENT) will notice the existence of the trigger file and begin to execute the transfer.

8.8 Summary

In this chapter, we looked at a hypothetical scenario for the JKHL company that involved a file transfer requirement with three distinct steps. Each step of the scenario was addressed using a different WebSphere MQ File Transfer Edition solution. The chapter also covered a real-world example showing how to use the WebSphere MQ File Transfer Edition *resource monitor* feature to integrate more complex integration and file transfer needs.

The file transfer steps that we describe in this chapter are:

1. Creating WebSphere MQ File Transfer Edition transfer definitions to automatically collect financial report files at the New York IT headquarter system.
2. Creating WebSphere MQ File Transfer Edition definitions to automatically trigger the execution of the business program to process the collected daily report files and to generate a daily consolidated data file.
3. Creating WebSphere MQ File Transfer Edition definitions to deliver the daily consolidated data file to the Chicago system for further processing.

In this chapter, we show how you can use WebSphere MQ File Transfer Edition to synchronize dependent, but separate, file transfer scenarios to yield a consistent fully automated solution.

9

Phase 3: Complex transfers

In this chapter, we describe the JKHL file transfer scenarios that require more complex solutions. We also discuss the requirement to reliably transfer large files. These scenarios include the use of fteAnt in combination with cURL to connect a SSH File Transfer Protocol (SFTP) server, the use of the resource monitor for triggering file transfers, and the use of integration with WebSphere Message Broker to accomplish data enrichment and automatic database updates.

The topics that we discuss in this chapter are:

9.1 Scenario overview

In the previous two chapters we described JKHL's incremental adoption of WebSphere File Transfer Edition (FTE) as a managed, file-based method for integrating their enterprise.

In Chapter 7, "Phase 1: Basic file transfers" on page 159, we describe the first phase of JKHL's application of FTE solutions to the solution of their file-based integration problems. In that phase, the basic features of the product are used to provide solutions for JKHL.

In Chapter 8, "Phase 2: Multi-step transfers" on page 191, we describe the second phase of JKHL's application of FTE solutions. In that phase, JKHL explores the use of more sophisticated—and more powerful—FTE features. The main features that are employed are the scheduling, triggering, and resource monitoring of file transfers.

Up to this point, JKHL has not yet explored:

- Using fteAnt scripting to enable complex JKHL business solutions
- Developing a mechanism for integrating their FTE-based file transfer approach with other integration products and with other file transfer protocols.

In this chapter, we discuss both of these topics. We also introduce solution scenarios that are more complex than in the preceding two chapters. Hopefully, this chapter gives you a view of the way that you can apply FTE to more complex real-world business solutions.

Specifically, we cover the following file transfer scenarios in this chapter:

- Distribute ad hoc travel package updates to all JKHL offices, including the office with only an SFTP server (a server with no FTE agent).
- Distribute software updates to all offices and recover from network failures during these 1 GB file transfers.
- Complete the daily financial reporting process, described in Chapter 8, "Phase 2: Multi-step transfers" on page 191, by automatically updating JKHL's corporate database.
- As one automated system, receive and process pricing updates, update the corporate relational database, and then distribute the updates all offices.

9.2 Distributing ad hoc travel package updates

From time-to-time, JKHL introduces new travel packages. These packages consist of multiple files and directories. These directories contain brochures, photographs, posters, costs, hotel, transportation, maps, and other information that is related to their new package. Each JKHL office use this information to promote the new offering. Each package is usually about 10 MB in size.

9.2.1 The requirement

In the past, JKHL prepared CDs for shipment to their worldwide offices, which was time consuming in both preparation and transport. Further, the packages did not always reach their destination.

JKHL plans to use FTE to electronically transfer these packages in one step and to have them reliably reach their destination.

Distributing a package to FTE-enabled offices

FTE can easily accomplish this delivery to the NY_FINANCIAL, WASHINGTON, CHICAGO, and MUMBAI machines. All of these machines are FTE-enabled.

Distributing a package to an office with an SFTP server

JKHL recently opened a new office in Tokyo. This office does not yet have FTE installed. The only secure way that JKHL can distribute files to TOKYO is to connect to TOKYO's SSH File Transfer Protocol SFTP server. Without an automated FTE-solution, a separate, manual file distribution process would be required.

9.2.2 The proposed solution

Fortunately, FTE provides a way to integrate the file transfer infrastructure with other file transfer protocols. The integration can be accomplished using the FTE-supplied Ant scripting tool. Starting with the release of FTE V7.0.1, Apache Ant is integrated with the product. This capability is invoked using the **`fteAnt`** command.

The **`fteAnt`** command allows for initiating transfers that include both FTE (over MQ) and transfers over other protocols. For this particular requirement, **`fteAnt`** calls the cURL program to transmit the package file to TOKYO over the SFTP protocol. We discussed cURL in detail in Chapter 14, "Integration" on page 349.

9.2.3 Setting up the transfer request

For JKHL's first package distribution test they send the package, AustralianOutback.zip to five of its offices, including the TOKYO machine. The size of the package is 13 MB.

Figure 9-1 shows what JKHL's package distribution solution looks like using **`fteAnt`**.

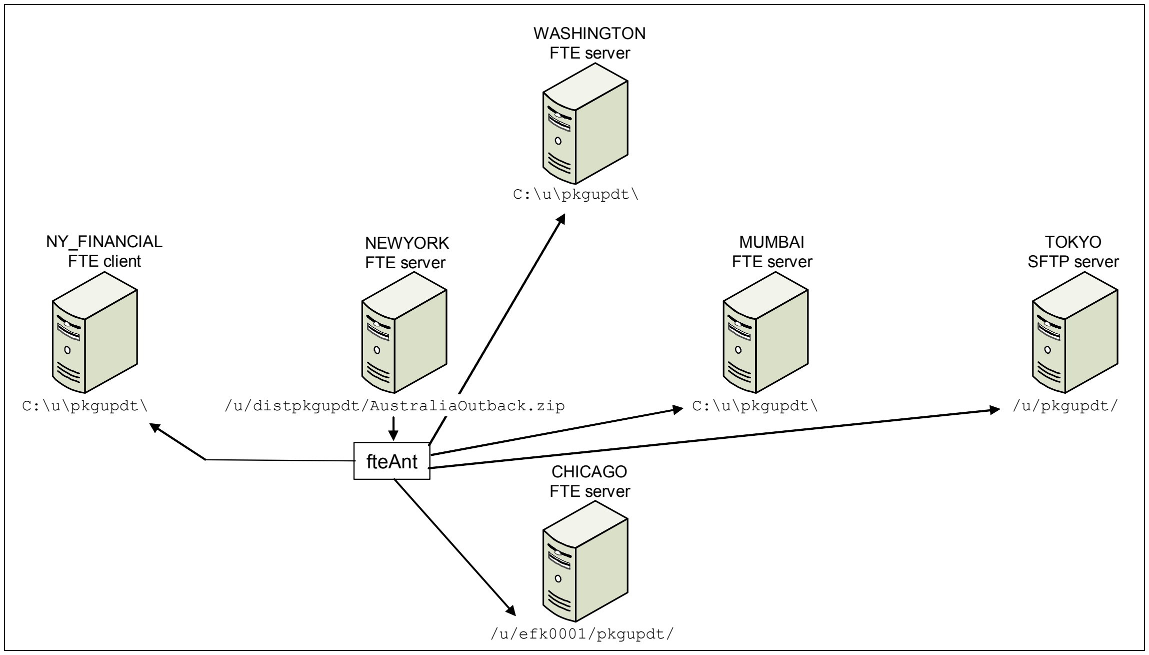


Figure 9-1 Package update distribution

The command for initiating the distribution of the AustralianOutback.zip package to all of the offices is run from the WASHINGTON machine, as shown in Example 9-1.

Example 9-1 Initiating the transfer from the NEWYORK command line

```
fteAnt -f pkg_dist.xml
```

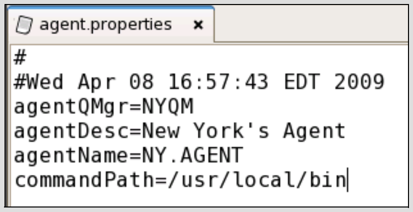
The pkg_dist.xml file, Example 9-2 on page 231, contains the specifications for the transfer.

Example 9-2 pkg_dist.xml

```
<?xml version='1.0'?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="copy">
<target name="copy">
<!-- To NY_FINANCIAL machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="NYFIN.AGENT@NYQM"
   rcproperty="copy.result">
   <fte:filespec srcfilespec="/u/distpkgupdt/AustralianOutback.zip"
         dstfile="c:\u\pkgupdt\AustralianOutback.zip"/>
   </fte:filecopy>
<!-- To MUMBAI machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="MUMBAI.AGENT@MUMBAIQM"
   rcproperty="copy.result">
   <fte:filespec srcfilespec="/u/distpkgupdt/AustralianOutback.zip"
            dstfile="c:\u\pkgupdt\AustralianOutback.zip"/>
   </fte:filecopy>
<!-- To WASHINGTON machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="WASH.AGENT@WASHQM"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/AustralianOutback.zip"
          dstfile="c:\u\pkgupdt\AustralianOutback.zip"/>
</fte:filecopy>
<!-- To CHICAGO machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="CHICAGO.AGENT@MQH1"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/AustralianOutback.zip"
          dstfile="/u/efk0001/pkgupdt/AustralianOutback.zip"/>
   </fte:filecopy>
<!-- To TOKYO machine -->
   <fte:call
      cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      agent="NY.AGENT@NYQM"
      rcproperty="call.rc">
      <fte:command command="curl" successrc="0" retrycount="0" retrywait="5">
         <fte:arg value="-T" />
    <fte:arg value="/u/distpkgupdt/AustralianOutback.zip" />
    <fte:arg value="--netrc" />
    <fte:arg value="sftp://tokyo/u/pkgupdt/AustralianOutback.zip" />
   <fte:arg value="--stderr" />
   <fte:arg value="curl.log" />
```

```
      </fte:command>
   </fte:call>
</target>
</project>
```

agent.properties: For the NY.AGENT to invoke the cURL program it must be authorized for that program's path in the agent's commandPath attribute, which is specified in the NY.AGENT's agent.properties file shown in Figure 9-2. This is discussed in 3.6.5, "agent.properties" on page 68.



```
#
#Wed Apr 08 16:57:43 EDT 2009
agentQMgr=NYQM
agentDesc=New York's Agent
agentName=NY.AGENT
commandPath=/usr/local/bin
```

Figure 9-2 Properties file

Each of the **filecopy** commands in Example 9-2 on page 231 transfers the package file using FTE over MQ. The **call** command causes the FTE NY.AGENT to execute the command in Example 9-3 to transfer the AustralianOutback.zip package to TOKYO using SFTP.

Example 9-3 Use of cURL

```
curl -T /u/distpkgupdt/AustralianOutback.zip --netrc
sftp://tokyo//u/pkgupdt/AustralianOutback.zip
```

On WASHINGTON, you can simplify the initiation of this Ant script by double-clicking the icon: Create a file, for example, pkg_updt.cmd, as shown in Figure 9-3. The file should contain the command shown in Example 9-4.

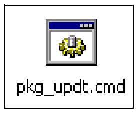


Figure 9-3 pkg_updt.cmd

Example 9-4 fteAnt command

```
cmd /k fteAnt -f pkg_dist.xml
```

9.2.4 Running the package distribution command manually

In this section, we provide an account of what happens when you run the command to distribute the travel package update.

1. From WASHINGTON, double click the **pkg_updt.cmd** file

   Example 9-5 appears in the console.

Example 9-5 Console output

```
C:\>cmd /k fteAnt -f c:\IBM\WMQFTE\ant\demos\pkg_dist.xml
5655-U80, 5724-R10 Copyright IBM Corp.  2008.  ALL RIGHTS RESERVED
BFGCL0193I: The request has been received by the source agent.
BFGCL0193I: The request has been received by the source agent.
BFGCL0193I: The request has been received by the source agent.
BFGCL0193I: The request has been received by the source agent.
BFGCL0217I: Issuing call request to agent NY.AGENT
BFGCL0193I: The request has been received by the source agent.
BFGCL0218I: Call request issued.  The request ID is:
414d51204e59514d2020202020202020f328f3490f240020
BFGCL0215I: call of command id
414d51204e59514d2020202020202020f328f3490f240020 from agent NY.AGENT
was successful.
```

2. The first four sections represent requests to the NY.AGENT to send the package update to NY_FINANCIAL, MUMBAI, WASHINGTON, and CHICAGO. The final section is a call request that invokes cURL to send the package to TOKYO.
3. When we check each of the five destination machines, each contains the AustralianOutback.zip package, including, for example, MUMBAI, as shown in Figure 9-4 and TOKYO, shown in Figure 9-5 on page 234.

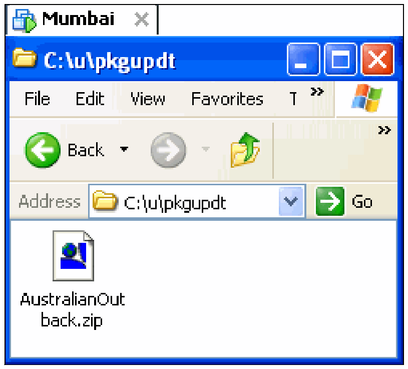


Figure 9-4 MUMBAI

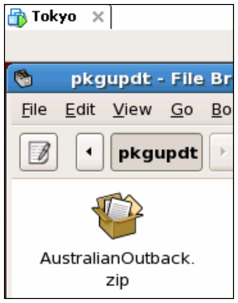
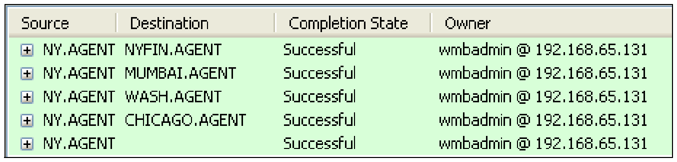


Figure 9-5 TOKYO

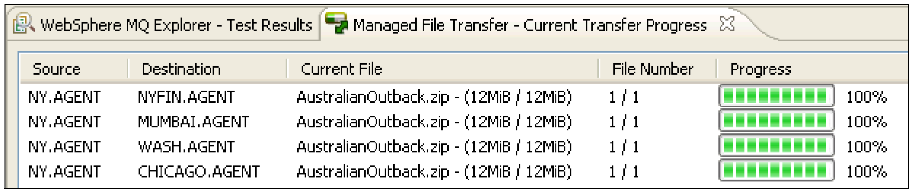
4. On WASHINGTON, if you look in MQ Explorer's transfer log (the coordination queue manager machine), you see that five transfer requests were initiated, as shown in Figure 9-6.



Source	Destination	Completion State	Owner
⊞ NY.AGENT	NYFIN.AGENT	Successful	wmbadmin @ 192.168.65.131
⊞ NY.AGENT	MUMBAI.AGENT	Successful	wmbadmin @ 192.168.65.131
⊞ NY.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131
⊞ NY.AGENT	CHICAGO.AGENT	Successful	wmbadmin @ 192.168.65.131
⊞ NY.AGENT		Successful	wmbadmin @ 192.168.65.131

Figure 9-6 Transfer log

The four FTE tasks are listed in the progress window shown in Figure 9-7.


WebSphere MQ Explorer - Test Results
Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress
NY.AGENT	NYFIN.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	MUMBAI.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	WASH.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	CHICAGO.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%

Figure 9-7 Progress view

5. Figure 9-8 on page 235 shows the progress log for the cURL task for TOKYO in the curl.log in the NY.AGENT directory.

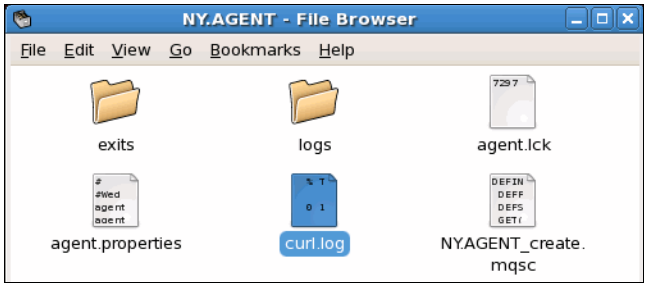


Figure 9-8 NY.AGENT

Example 9-6 shows the content of that cURL log.

Example 9-6 cURL log

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0 12.7M    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  7 12.7M    0     0    7 1040k      0  1074k  0:00:12 --:--:--  0:00:12 1319k
 17 12.7M    0     0   17 2304k      0  1170k  0:00:11  0:00:01  0:00:10 1288k
 27 12.7M    0     0   27 3632k      0  1223k  0:00:10  0:00:02  0:00:08 1302k
 37 12.7M    0     0   37 4912k      0  1237k  0:00:10  0:00:03  0:00:07 1296k
 46 12.7M    0     0   46 6128k      0  1233k  0:00:10  0:00:04  0:00:06 1279k
 56 12.7M    0     0   56 7408k      0  1241k  0:00:10  0:00:05  0:00:05 1273k
 66 12.7M    0     0   66 8688k      0  1246k  0:00:10  0:00:06  0:00:04 1276k
 76 12.7M    0     0   76  9.7M      0  1255k  0:00:10  0:00:07  0:00:03 1273k
 86 12.7M    0     0   86 11.0M      0  1261k  0:00:10  0:00:08  0:00:02 1280k
 96 12.7M    0     0   96 12.2M      0  1263k  0:00:10  0:00:09  0:00:01 1292k
100 12.7M    0     0  100 12.7M      0  1259k  0:00:10  0:00:10 --:--:-- 1283k
100 12.7M    0     0  100 12.7M      0  1259k  0:00:10  0:00:10 --:--:-- 1259k
```

9.2.5 Running the package distribution command automatically using an FTE resource monitor

In Chapter 8, “Phase 2: Multi-step transfers” on page 191, we describe how we use FTE resource monitors to perform file transfers as their associated task. We can also configure the resource monitor to invoke other commands, which includes executable programs, Ant scripts, or JCL. To do this, you must edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties. 8.6.1, “Setting up the New York Agent using the command line” on page 219, shows an example of this capability, where we used a managedCall transfer specification to call a program that consolidated daily financial reports.

In this section, we use a managedCall in conjunction with the resource monitor to automatically execute the Ant script, pkg_dist.xml, that we ran manually in 9.2.4, “Running the package distribution command manually” on page 233.

Here is an overview of the steps that we followed to set up an automatic package distribution:

1. Put a copy of pkg_dist.xml in a script directory on NEWYORK. (The NY.AGENT initiates the file transfer.)
2. Add pkg_dist.xml’s directory path to the commandPath parameter of the NY.AGENT’s agent.properties file.
3. Create the managedCall XML specification, mgd_call.xml. This command file is triggered to call pkg_dist.xml. Include in this command file the variable, packageFile. This variable is assigned a package name to be transferred.
4. In pkg_dist.xml, replace the hard-coded package names with the variable, packageFile.
5. Test mgd_call.xml and the modified pkg_dist.xml that it calls using RFHUTILC.
6. Specify what triggers the managedCall.
7. Define, then create, the resource monitor that watches for the trigger and initiates the managedCall.
8. Test the monitor.

Putting a copy of pkg_dist.xml in a directory on NEWYORK

We put a copy of pkg_dist.xml in the /home/wmbadmin/IBM/scripts directory on NEWYORK.

Adding pkg_dist.xml’s directory path to the agent.properties file

Because this Ant script will be run from the /home/wmbadmin/IBM/scripts directory on NEWYORK, we added this path to the NY.AGENT’s agent.properties file, as shown in Example 9-7 on page 237.

Example 9-7 The NY.AGENT agent.properties file after adding the pkg_dist.xml path

```
#Tue May 05 13:03:30 EDT 2009
agentQMgr=NYQM
agentDesc=New York's agent
agentName=NY.AGENT
commandPath=/home/wmbadmin/IBM/scripts:/usr/local/bin
```

Note: An agent can execute only those scripts/executables that are located within a path that is specified in the commandPath parameter of the agent.Properties file. The path, /usr/local/bin, is the path to the cURL executable that is called by pkg_dist.xml.

Creating the managedCall XML specification, mgd_call.xml

We created the managedCall specification, shown in Example 9-8, to call the pkg_dist.xml Ant script.

Example 9-8 mgd_call.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
   <originator>
      <hostName>newyork</hostName>
      <userID>wmbadmin</userID>
   </originator>
   <agent QMgr="NYQM" agent="NY.AGENT"/>
   <reply QMGR="NYQM">reply</reply>
   <transferSet priority="1">
      <call>
         <command name="pkg_dist.xml" type="antscript"
            retryCount="0" retryWait="0" successRC="0">
         <property name="packageFile"
            value="AustralianOutback.zip"/>
         </command>
      </call>
   </transferSet>
</managedCall>
</request>
```

The variable, packageFile, was included in the mgd_call.xml so that the name of the package file to be transferred could be assigned to this variable one time, and then passed to the called Ant script, pkg_dist.xml.

We saved the XML, shown in Example 9-8 on page 237 as a file named, mgd_call.xml in the /home/wmbadmin/IBM/scripts directory on NEWYORK.

Replacing the hard-coded package names in pkg_dist.xml

In the Ant script, pkg_dist.xml, the five hard-coded package names were replaced by the variable, packageFile. The value of this variable is assigned in mgd_call.xml at runtime, as shown in Example 9-9.

Example 9-9 Revised pkg_dist.xml

```
<?xml version='1.0'?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs">
<!-- To NY_FINANCIAL machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="NYFIN.AGENT@NYQM"
   rcproperty="copy.result">
   <fte:filespecsrcfilespec="/u/distpkgupdt/${packageFile}"
         dstfile="c:\u\pkgupdt\${packageFile}"/>
   </fte:filecopy>
<!-- To MUMBAI machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="MUMBAI.AGENT@MUMBAIQM"
   rcproperty="copy.result">
   <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
            dstfile="c:\u\pkgupdt\${packageFile}"/>
   </fte:filecopy>
<!-- To WASHINGTON machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="WASH.AGENT@WASHQM"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
          dstfile="c:\u\pkgupdt\${packageFile}"/>
</fte:filecopy>
<!-- To CHICAGO machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="CHICAGO.AGENT@MQH1"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
          dstfile="/u/efk0001/pkgupdt/${packageFile}"/>
   </fte:filecopy>
<!-- To TOKYO machine -->
```

```
   <fte:call
      cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      agent="NY.AGENT@NYQM"
      rcproperty="call.rc">
      <fte:command command="curl" successrc="0" retrycount="0" 
retrywait="5">
         <fte:arg value="-T" />
    <fte:arg value="/u/distpkgupdt/${packageFile}" />
    <fte:arg value="--netrc" />
    <fte:arg value="sftp://tokyo//u/pkgupdt/${packageFile}" />
   <fte:arg value="--stderr" />
   <fte:arg value="curl.log" />
      </fte:command>
   </fte:call>
</project>
```

Testing mgd_call.xml and the revised pkg_dist.xml

Before using mgd_call.xml and our revised pkg_dist.xml with a resource monitor, we found it useful to:

1. Test the mgd_call.xml to make sure that it correctly calls our Ant script, pkg_dist.xml.
2. Ensure that our revised pkg_dist.xml script still works, for example, distributes the specified package file.

To perform this test, we used the free utility RFHUTILC to trigger mgd_call.xml. This utility is available as an IBM SupportPac at:

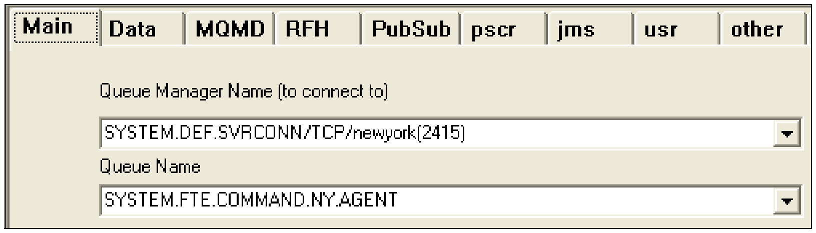
http://www-01.ibm.com/support/docview.wss?uid=swg24000637

This test has a side-benefit of helping to understand how FTE works.

We installed RFHUTILC on the WASHINGTON machine, and put a copy of mgd_call.xml in the c:\ directory. The copy of pkg_dist.xml that will be called by mgd_call.xml is located in the /home/wmbadmin/IBM/scripts directory on NEWYORK.

To test the XML:

1. Open RFHUTILC, and enter the queue manager name and command queue name of NY.AGENT, as shown in Figure 9-9 on page 240.



Queue Manager Name (to connect to)

SYSTEM.DEF.SVRCONN/TCP/newyork(2415)

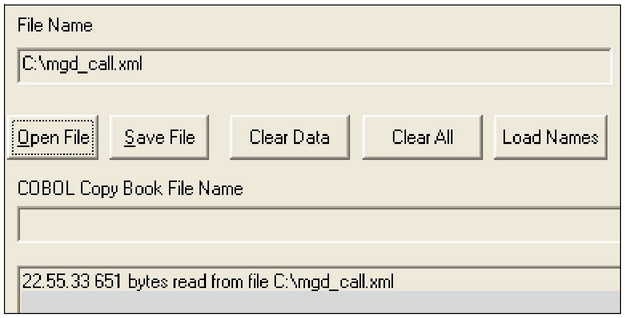
Queue Name

SYSTEM.FTE.COMMAND.NY.AGENT

Figure 9-9 Enter queue manager and queue name

Note: In Figure 9-9 on page 240, to use "newyork" as the machine name, its IP address must be defined in your host's file. If it is not, use NEWYORK's IP address instead.

2. Click **Open File**, shown in Figure 9-10, and then open the XML specification file to be tested, mgd_call.xml.



File Name

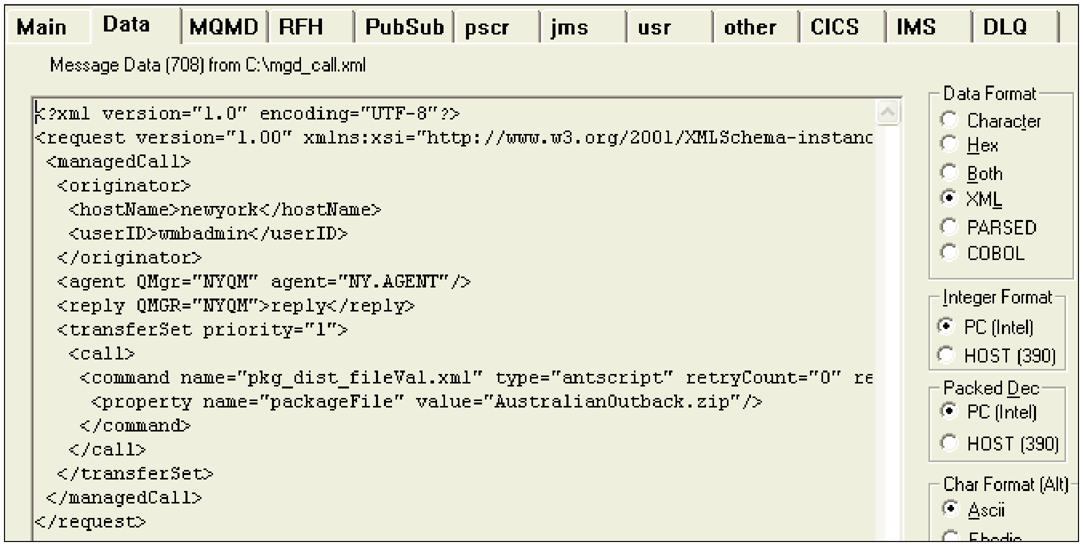
C:\mgd_call.xml

COBOL Copy Book File Name

22.55.33 651 bytes read from file C:\mgd_call.xml

Figure 9-10 Open the mgd_call.xml file

3. Click the **Data** tab, and select the **XML** option, Figure 9-11 on page 241, to confirm that the correct file was opened.



Message Data (708) from C:\mgd_call.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instanc
 <managedCall>
  <originator>
   <hostName>newyork</hostName>
   <userID>wmbadmin</userID>
  </originator>
  <agent QMgr="NYQM" agent="NY.AGENT"/>
  <reply QMGR="NYQM">reply</reply>
  <transferSet priority="1">
   <call>
    <command name="pkg_dist_fileVal.xml" type="antscript" retryCount="0" re
     <property name="packageFile" value="AustralianOutback.zip"/>
    </command>
   </call>
  </transferSet>
 </managedCall>
</request>
```

Data Format: Character, Hex, Both, [✓ XML], PARSED, COBOL

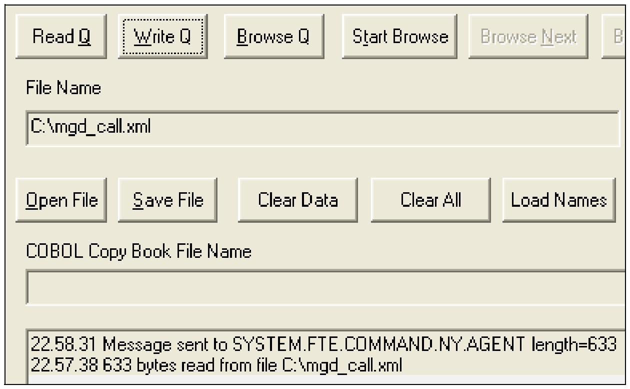
Integer Format: [✓ PC (Intel)], HOST (390)

Packed Dec: [✓ PC (Intel)], HOST (390)

Char Format (Alt): [✓ Ascii], Ebcdic

Figure 9-11 Confirm that the correct file has been opened

4. Click **WriteQ**, Figure 9-12, to send the XML command to the NY.AGENT command queue. FTE then runs the command as though it was sent by the resource monitor.



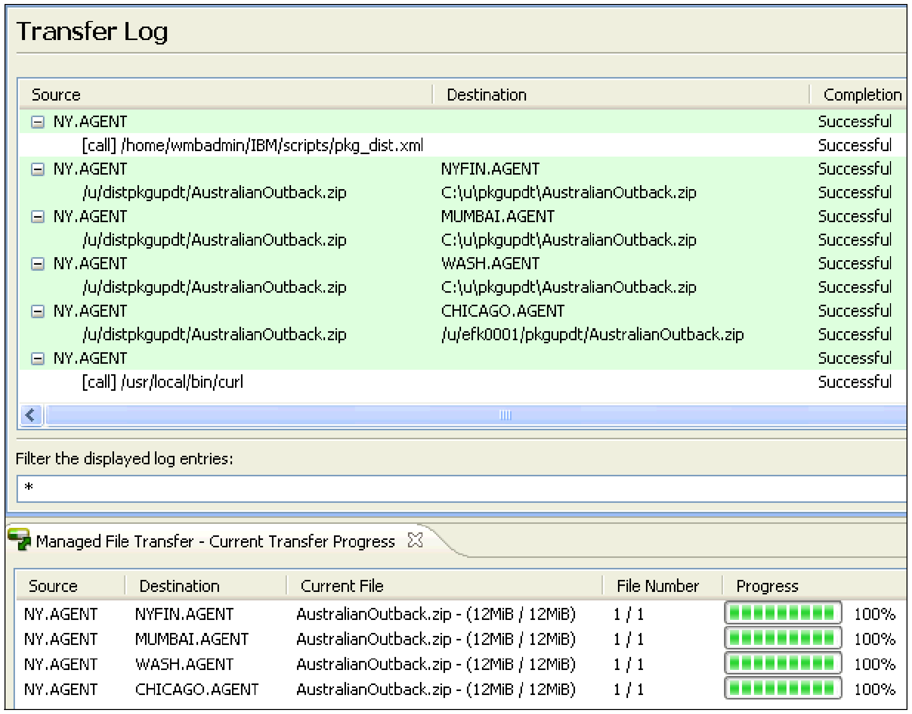
File Name: C:\mgd_call.xml

COBOL Copy Book File Name:

```
22.58.31 Message sent to SYSTEM.FTE.COMMAND.NY.AGENT length=633
22.57.38 633 bytes read from file C:\mgd_call.xml
```

Figure 9-12 Send XML to NY.AGENT command queue

5. On WASHINGTON, check the transfer log in MQ Explorer. The result should be similar to Figure 9-14 on page 243.


Transfer Log

Source	Destination	Completion
⊟ NY.AGENT		Successful
[call] /home/wmbadmin/IBM/scripts/pkg_dist.xml		Successful
⊟ NY.AGENT	NYFIN.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	MUMBAI.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	WASH.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	CHICAGO.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	/u/efk0001/pkgupdt/AustralianOutback.zip	Successful
⊟ NY.AGENT		Successful
[call] /usr/local/bin/curl		Successful

Filter the displayed log entries:

*

Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress
NY.AGENT	NYFIN.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	MUMBAI.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	WASH.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	CHICAGO.AGENT	AustralianOutback.zip - (12MiB / 12MiB)	1 / 1	100%

Figure 9-13 Result of RFHUTIL test as shown in MQ Explorer on WASHINGTON

6. Check the transfer destination directories to see if the packages were distributed as expected.
7. Delete AustraliaOutback.zip from all destination directories, for example, from c:\u\pkgupdt on MUMBAI.

Specifying what triggers the managedCall

We modified the design of the system, as shown in Figure 9-1 on page 230, so that after a new package is created in the /u/distpkgupdt directory, a trigger file, pkgupdt.trigger, is also created, as shown in Figure 9-14 on page 243.

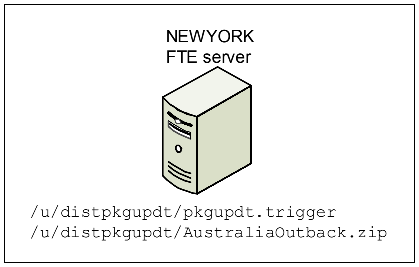

NEWYORK
FTE server
/u/distpkgupdt/pkgupdt.trigger
/u/distpkgupdt/AustraliaOutback.zip

Figure 9-14 After a new package is created, a trigger file is added

In our new design, the presence of a this trigger file initiates an automatic package distribution.

Trigger file: Using a trigger file (separate from the file-to-be-transferred) is an important best practice. A trigger file's arrival in a source directory *after* the arrival of the file-to-be-transferred ensures that a file is completely written before a transfer to a destination directory begins.

For now, we place only the package, AustraliaOutback.zip, in the /u/distpkgupdt directory. We will add the trigger file later.

Defining and creating the resource monitor

We use the **fteCreateMonitor** command to create a resource monitor on the NY.AGENT. This resource monitor checks the /u/distpkgupdt directory on NEWYORK. When a new pkgupdt.trigger file is created in this directory, the resource monitor executes the mgd_call.xml. Example 9-8 on page 237 shows the command for creating the monitor, pkg_monitor.

On NEWYORK, at the command line in the /home/wmbadmin/IBM/scripts directory, run the command in Example 9-10.

Example 9-10 Command for creating the monitor, pkg_monitor

```
fteCreateMonitor
      -ma NY.AGENT
      -mm NYQM
      -md /u/distpkgupdt
      -mn pkg_monitor
      -mt mgd_call.xml
      -pi 10
      -pu seconds
      -tr match,pkgupdt.trigger
```

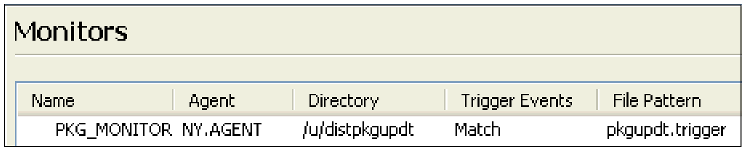
Carriage returns: For clarity, the **fteCreateMonitor** command in Example 9-10 displays its individual parameters on separate lines using carriage returns. In the actual command, all parameters must be on the same line, not using carriage returns.

Example 9-11 shows what appears in the console when the **fteCreateMonitor** command is run.

Example 9-11 Console output

```
[wmbadmin@newyork scripts]$ fteCreateMonitor -ma NY.AGENT -mm NYQM -md
/u/distpkgupdt -mn pkg_monitor -mt mgd_call.xml -pi 10 -pu seconds -tr
match,pkgupdt.trigger
5655-U80, 5724-R10 Copyright IBM Corp.  2008, 2009.  ALL RIGHTS
RESERVED
BFGCL0188I: The requested monitor has successfully completed with a
transfer id of 414d51204e59514d2020202020202020c007034a76e80020.
[wmbadmin@newyork scripts]$
```

On WASHINGTON, the monitor is displayed in MQ Explorer, as shown in Figure 9-15.


Monitors

Name	Agent	Directory	Trigger Events	File Pattern
PKG_MONITOR	NY.AGENT	/u/distpkgupdt	Match	pkgupdt.trigger

Figure 9-15 Configuration queue manager's view of pkg_monitor

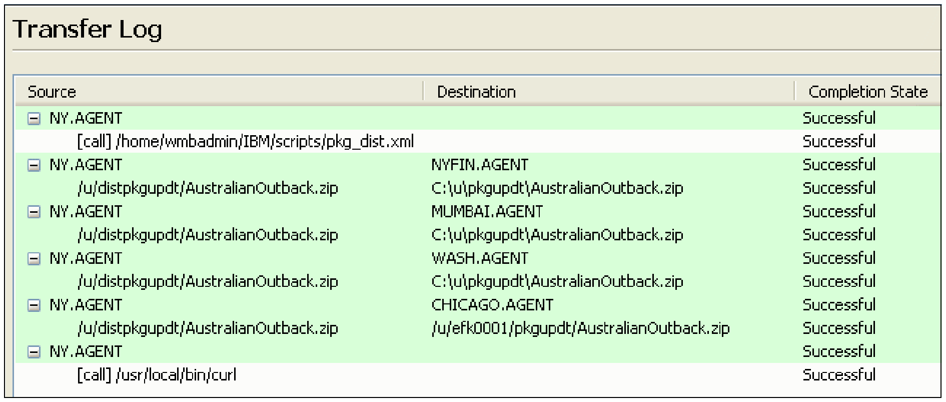
Testing the monitor

On NEWYORK, we changed to the /u/distpkgupdt directory. Next, we made sure that there is a copy of AustraliaOutback.zip in this directory. Finally, we created a trigger file by entering the command in Example 9-12.

Example 9-12 Create the trigger file

```
[wmbadmin@newyork distpkgupdt]$ touch pkgupdt.trigger
```

Creating the trigger file caused the resource monitor to automatically initiate the mgd_call.xml command file. This file, in turn, called the Ant script, pkg_dist.xml, that distributed the new package to all JKHL machines, including through SFTP to TOKYO using cURL. The transfer log in Figure 9-16 on page 245 shows the sequence of events that the creating a new trigger file initiated.


Transfer Log

Source	Destination	Completion State
⊟ NY.AGENT		Successful
[call] /home/wmbadmin/IBM/scripts/pkg_dist.xml		Successful
⊟ NY.AGENT	NYFIN.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	MUMBAI.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	WASH.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	C:\u\pkgupdt\AustralianOutback.zip	Successful
⊟ NY.AGENT	CHICAGO.AGENT	Successful
/u/distpkgupdt/AustralianOutback.zip	/u/efk0001/pkgupdt/AustralianOutback.zip	Successful
⊟ NY.AGENT		Successful
[call] /usr/local/bin/curl		Successful

Figure 9-16 Transfer log after automatic package distribution

9.2.6 Making automatic package distribution work for any package

In the previous section, we used the FTE resource monitor to automatically distribute a package file. This transfer was triggered by the arrival of the trigger file, pkgupdt.trigger. However, a serious limitation to our approach is that the name of the package file to be distributed was hard-coded in mgd_call.xml.

We therefore made some minor modifications to our design to allow JKHL to trigger the distribution of *any* package file.

Here is an overview of the steps we used to accomplish generic package file distribution:

1. Change the design for the trigger name. The format is now [package name].trigger.
2. Change the trigger parameter in the resource monitor definition to *.trigger for example, use a wildcard so that the resource monitor is triggered by any .trigger file.
3. Modify mgd_call.xml to obtain the current package name from its associated trigger name.
4. Create the new resource monitor.
5. Test the monitor.

Changing the design for the trigger name

We modified the design of the system, Figure 9-13 on page 242, so that after a new package is created in the /u/distpkgupdt directory, a trigger file, [package name].trigger, is also created, as shown in Figure 9-17 on page 246.

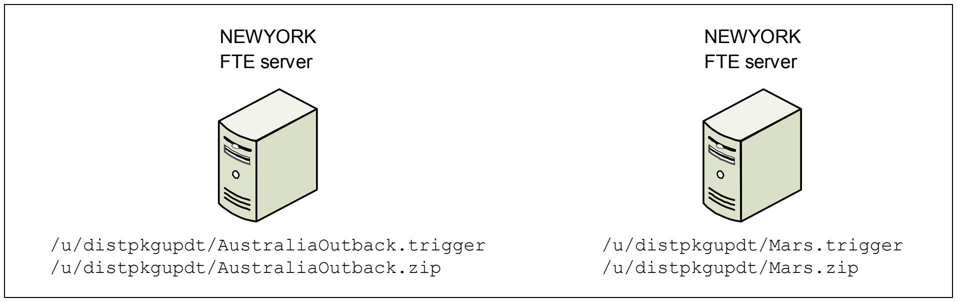

NEWYORK
FTE server
/u/distpkgupdt/AustraliaOutback.trigger
/u/distpkgupdt/AustraliaOutback.zip
NEWYORK
FTE server
/u/distpkgupdt/Mars.trigger
/u/distpkgupdt/Mars.zip

Figure 9-17 Examples of trigger names using the new design

In our new design, the presence of a [package name].trigger file in the /u/distpkgupdt directory initiates an automatic package distribution of the corresponding [package name].zip file.

Changing the trigger parameter in the resource monitor definition

In our new design, the resource monitor must be triggered by any trigger file in the form, [package name].trigger. Previously, in our resource monitor definition, we used the fixed trigger name, pkgupdt.trigger. We now use a wildcard to specify the trigger file as *.trigger.

Our new resource monitor definition, shown in Example 9-13, creates a resource monitor that monitors the /u/distpkgupdt directory for *any* new trigger file in the form [package name].trigger, and then executes mgd_call.xml.

Example 9-13 Resource monitor definition using a wildcard in the trigger name

```
fteCreateMonitor
      -ma NY.AGENT
      -mm NYQM
      -md /u/distpkgupdt
      -mn pkg_monitor2
      -mt mgd_call.xml
      -pi 10
      -pu seconds
      -tr match,*.trigger
```

Carriage Returns: For clarity, the **fteCreateMonitor** command in Example 9-13 displays its individual parameters on separate lines using carriage returns. In the actual command, all parameters must be on the same line (no carriage returns).

Modifying mgd_call.xml

Recall that when using our FTE resource monitor, the arrival of a trigger file causes mgd_call.xml to call the Ant script, pkg_dist.xml. This Ant script uses the value of the variable, packageFile, to identify the name of the package to be distributed.

In our previous design, the value of packageFile was hard-coded, as shown in Figure 9-18.

```
<property name="packageFile"
   value="AustralianOutback.zip"/>
```

Figure 9-18 Previous design: packageFile definition is hard-coded

In our new design, we changed the definition of packageFile, as shown in Figure 9-19.

```
<property name="packageFile"
   value="${FileName{token=1}{separator=.}}.zip"/>
```

Figure 9-19 New design: packageFile definition derived from any trigger name

Our new generic design allows us to use *any* trigger name. We made use of the fact that the resource monitor always stores the value of the current trigger file in the variable, "${FileName}". Therefore, the value of packageFile can be derived, on-the-fly, from the current value of [package name].trigger.

The variable, "${FileName}", refers to the *complete* trigger name, for example, Mars.trigger. Whereas ${FileName{token=1}{separator=.}} refers only to the portion of the trigger name that is to the left of the dot, for example, "Mars".

Therefore, a trigger named Mars.trigger results in the value of packageFile being Mars.zip. The called Ant script, pkg_dist.xml, because it uses the value packageFile, distributes the package, Mars.zip.

In general then, using this new variable definition, a trigger file in the form, [package name].trigger, causes the value of packageFile to be, [package name].zip, which results in the called Ant script, pkg_dis.xml, distributing the file, [package name].zip.

Note: For further information, see “Customizing tasks with variable substitution”, in the FTE Information Center at the following Web address:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/variable_substitution.htm

We saved the modified mgd_call.xml file as mgd_call2.xml, as shown in Example 9-14.

Example 9-14 mgd_call2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
   <originator>
      <hostName>newyork</hostName>
      <userID>wmbadmin</userID>
   </originator>
   <agent QMgr="NYQM" agent="NY.AGENT"/>
   <reply QMGR="NYQM">reply</reply>
   <transferSet priority="1">
      <call>
         <command name="pkg_dist.xml" type="antscript"
            retryCount="0" retryWait="0" successRC="0">
         <property name="packageFile"
            value="${FileName{token=1}{separator=.}}.zip"/>
         </command>
      </call>
   </transferSet>
</managedCall>
</request>
```

No changes are required for the called Ant script, pkg_dist.xml. It already uses the current value of packageFile, as shown in Example 9-15.

Example 9-15 pkg_dist.xml

```
<?xml version='1.0'?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs">
<!-- To NY_FINANCIAL machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="NYFIN.AGENT@NYQM"
   rcproperty="copy.result">
   <fte:filespecsrcfilespec="/u/distpkgupdt/${packageFile}"
         dstfile="c:\u\pkgupdt\${packageFile}"/>
   </fte:filecopy>
<!-- To MUMBAI machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="MUMBAI.AGENT@MUMBAIQM"
   rcproperty="copy.result">
   <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
            dstfile="c:\u\pkgupdt\${packageFile}"/>
   </fte:filecopy>
<!-- To WASHINGTON machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="WASH.AGENT@WASHQM"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
          dstfile="c:\u\pkgupdt\${packageFile}"/>
</fte:filecopy>
<!-- To CHICAGO machine -->
   <fte:filecopy cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      src="NY.AGENT@NYQM" dst="CHICAGO.AGENT@MQH1"
      rcproperty="copy.result">
      <fte:filespec srcfilespec="/u/distpkgupdt/${packageFile}"
          dstfile="/u/efk0001/pkgupdt/${packageFile}"/>
   </fte:filecopy>
<!-- To TOKYO machine -->
   <fte:call
      cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
      agent="NY.AGENT@NYQM"
      rcproperty="call.rc">
      <fte:command command="curl" successrc="0" retrycount="0"
retrywait="5">
         <fte:arg value="-T" />
    <fte:arg value="/u/distpkgupdt/${packageFile}" />
    <fte:arg value="--netrc" />
    <fte:arg value="sftp://tokyo//u/pkgupdt/${packageFile}" />
   <fte:arg value="--stderr" />
   <fte:arg value="curl.log" />
      </fte:command>
   </fte:call>
</project>
```

Creating the new resource monitor

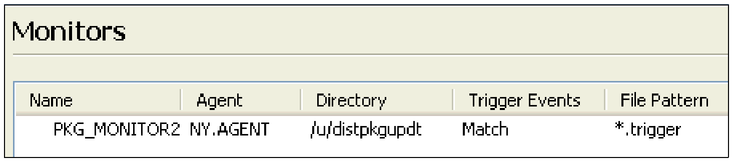
Using the monitor definition file in Figure 9-13 on page 242, we created a new resource monitor, pkg_monitor2, which we show in Example 9-16 on page 250.

Example 9-16 Command to create pkg_monitor2

```
[wmbadmin@newyork scripts]$ fteCreateMonitor -ma NY.AGENT -mm NYQM -md
/u/distpkgupdt -mn pkg_monitor2 -mt mgd_call2.xml -pi 10 -pu seconds
-tr match,*.trigger
5655-U80, 5724-R10 Copyright IBM Corp.  2008, 2009.  ALL RIGHTS
RESERVED
```

BFGCL0188I: The request to create a monitor has been submitted with a request id of
414d51204e59514d2020202020202020ab0a444a5cbc0520.

On WASHINGTON, the monitor appears in MQ Explorer, as shown in Figure 9-20.


Monitors

Name	Agent	Directory	Trigger Events	File Pattern
PKG_MONITOR2	NY.AGENT	/u/distpkgupdt	Match	*.trigger

Figure 9-20 Configuration queue manager's view of pkg_monitor2

Testing the monitor

In this section, we test the monitor using two tests.

Test 1

On NEWYORK, we placed a copy of the test travel package, Mars.zip, in the /u/distpkgupdt directory. Next, we created a trigger file by entering the command in Example 9-17.

Example 9-17 Create the trigger file

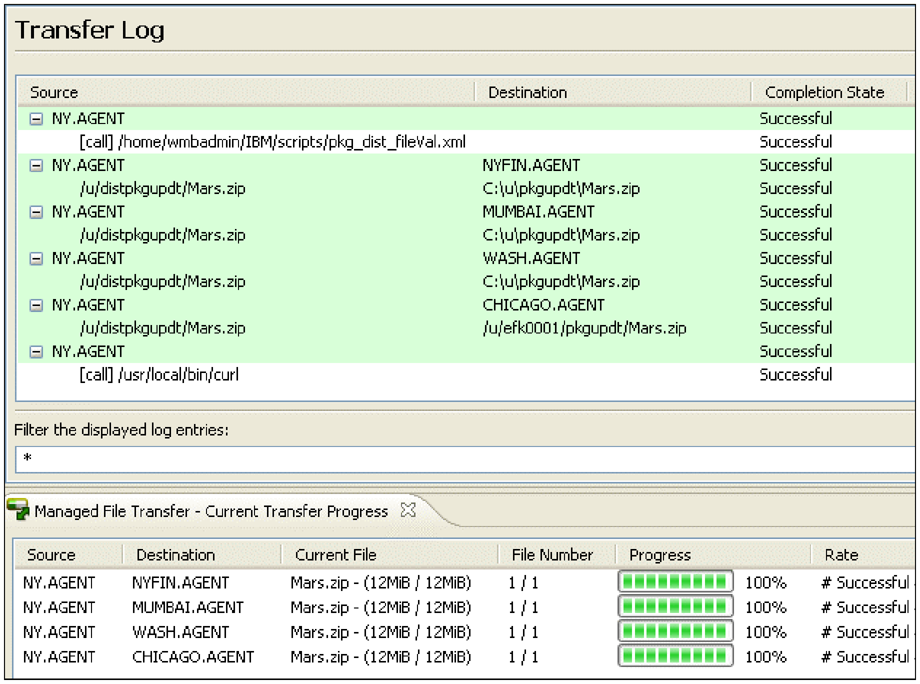
```
[wmbadmin@newyork distpkgupdt]$ touch Mars.trigger
```

Creating the trigger file caused the resource monitor to initiate the mgd_call.xml command file. This command file:

1. Took the name of the trigger file (to the left of the dot), for example, “Mars”.
2. Appended “Mars” to “.zip”.
3. Assigned the value, “Mars.zip” to the variable, packageFile.

The command file then called the Ant script, pkg_dist.xml. It distributed the new package to all JKHL machines, including through SFTP to TOKYO using cURL.

The transfer log in Figure 9-21 on page 251 shows the sequence of events that creating a new trigger file initiated.


Transfer Log

Source	Destination	Completion State
⊟ NY.AGENT		Successful
[call] /home/wmbadmin/IBM/scripts/pkg_dist_fileVal.xml		Successful
⊟ NY.AGENT	NYFIN.AGENT	Successful
/u/distpkgupdt/Mars.zip	C:\u\pkgupdt\Mars.zip	Successful
⊟ NY.AGENT	MUMBAI.AGENT	Successful
/u/distpkgupdt/Mars.zip	C:\u\pkgupdt\Mars.zip	Successful
⊟ NY.AGENT	WASH.AGENT	Successful
/u/distpkgupdt/Mars.zip	C:\u\pkgupdt\Mars.zip	Successful
⊟ NY.AGENT	CHICAGO.AGENT	Successful
/u/distpkgupdt/Mars.zip	/u/efk0001/pkgupdt/Mars.zip	Successful
⊟ NY.AGENT		Successful
[call] /usr/local/bin/curl		Successful

Filter the displayed log entries:
*

Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress	Rate
NY.AGENT	NYFIN.AGENT	Mars.zip - (12MiB / 12MiB)	1 / 1	100%	# Successful
NY.AGENT	MUMBAI.AGENT	Mars.zip - (12MiB / 12MiB)	1 / 1	100%	# Successful
NY.AGENT	WASH.AGENT	Mars.zip - (12MiB / 12MiB)	1 / 1	100%	# Successful
NY.AGENT	CHICAGO.AGENT	Mars.zip - (12MiB / 12MiB)	1 / 1	100%	# Successful

Figure 9-21 Test of Mars.zip transfer

Test 2

On NEWYORK, we placed a copy of the test travel package, DarjeelingEstates.zip, in the /u/distpkgupdt directory. Next, we created a trigger file by entering the command in Example 9-18.

Example 9-18 Create the trigger file

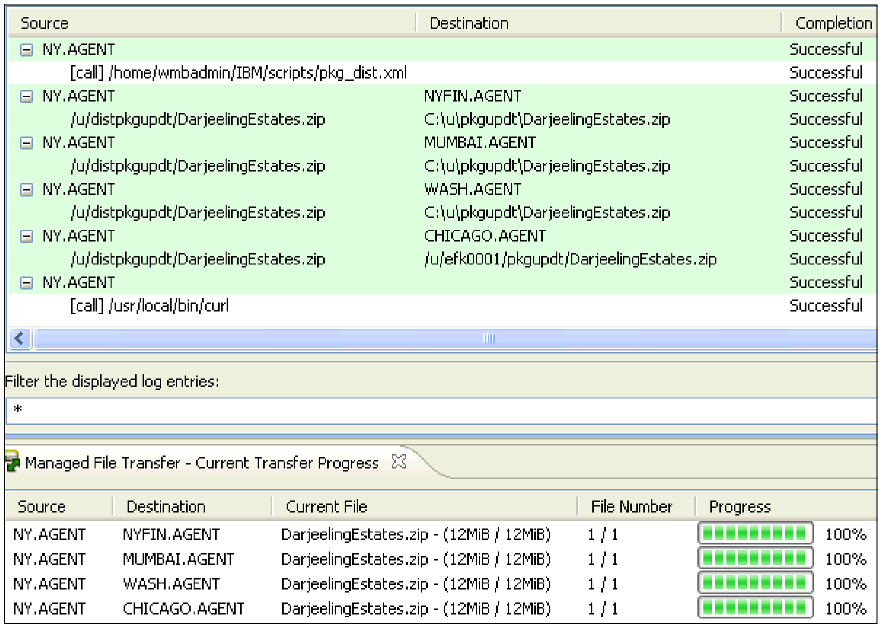
```
[wmbadmin@newyork distpkgupdt]$ touch DarjeelingEstates.trigger
```

Creating the trigger file caused the resource monitor to initiate the mgd_call.xml command file. This command file:

1. Took the name of the trigger file (to the left of the dot), for example, “DarjeelingEstates”.
2. Appended “DarjeelingEstates” to “.zip”.
3. Assigned the value, “DarjeelingEstates.zip” to the variable, packageFile.

The command file then called the Ant script, pkg_dist.xml. It distributed the new package to all JKHL machines, including through SFTP to TOKYO using cURL.

The transfer log, Figure 9-22, shows the sequence of events that the creating a new trigger file initiated.



Source	Destination	Completion
NY.AGENT		Successful
[call] /home/wmbadmin/IBM/scripts/pkg_dist.xml		Successful
NY.AGENT	NYFIN.AGENT	Successful
/u/distpkgupdt/DarjeelingEstates.zip	C:\u\pkgupdt\DarjeelingEstates.zip	Successful
NY.AGENT	MUMBAI.AGENT	Successful
/u/distpkgupdt/DarjeelingEstates.zip	C:\u\pkgupdt\DarjeelingEstates.zip	Successful
NY.AGENT	WASH.AGENT	Successful
/u/distpkgupdt/DarjeelingEstates.zip	C:\u\pkgupdt\DarjeelingEstates.zip	Successful
NY.AGENT	CHICAGO.AGENT	Successful
/u/distpkgupdt/DarjeelingEstates.zip	/u/efk0001/pkgupdt/DarjeelingEstates.zip	Successful
NY.AGENT		Successful
[call] /usr/local/bin/curl		Successful

Filter the displayed log entries:

*

Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress
NY.AGENT	NYFIN.AGENT	DarjeelingEstates.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	MUMBAI.AGENT	DarjeelingEstates.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	WASH.AGENT	DarjeelingEstates.zip - (12MiB / 12MiB)	1 / 1	100%
NY.AGENT	CHICAGO.AGENT	DarjeelingEstates.zip - (12MiB / 12MiB)	1 / 1	100%

Figure 9-22 Test of DarjeelinEstates.zip transfer

9.2.7 Discussion of the results

Using the **fteAnt command**, JKHL successfully distributed a travel update package to *all* of their offices. Part of that success was to send the package to an office that, for now, only supports SFTP, which allows JKHL to accomplish an incremental adoption of FTE as a managed file transfer solution.

Another JKHL requirements was to accomplish secure file transfers.

As shown in Example 9-15 on page 248, **fteAnt** used two methods to securely transfer JKHL's new travel package:

1. Transfers to their FTE-enabled machines (NY_FINANCIAL, MUMBAI, WASHINGTON, and CHICAGO) occurred over MQ with Secure Socket Layer (SSL) enabled, which we show as the **fteAnt** *filecopy* function.
2. Transfers to the office that did not support FTE (TOKYO machine) were accomplished using the SSH File Transfer Protocol, which we show as the **fteAnt** *call* function.

Building upon the success of this test, JKHL can use the **`fteAnt`** command to solve other file transfer problems. As an example, JKHL could, in one step, use the **`fteAnt`** command to:

- Transfer many different files, from many locations, to one destination.
- Transfer many files, from many locations, to many destinations.
- Both send and receive files in the same command set.
- Include transitional support for additional protocols. Using a call to cURL, support for the following protocols could also be included: FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS, and FILE.
- Call any authorized executable to accomplish integration with other file-transfer approaches.

9.3 Completing the financial reporting system

Creating the JKHL daily financial reporting system using FTE managed file transfer capabilities was nearly completed during the phase that we described in Chapter 8, “Phase 2: Multi-step transfers” on page 191. However, there are several parts of that system that we must complete:

- Since the initial creation of the financial reporting system, JKHL acquired an additional office, Tokyo. This office does not have FTE installed. Its only method of interfacing with the JKHL infrastructure is using its SFTP server.

  Therefore, we must develop a way to automatically retrieve Tokyo’s daily financial reports.
- The back end process of the financial reporting system, the updating of the corporate database, is still accomplished by manual data entry to a CICS® database update application. JKHL wants to automate that process and integrate it with the FTE-based financial reporting system.

In this section, we discuss how to achieve these goals.

9.3.1 Retrieving TOYKO daily financial reports using SFTP

Retrieving daily financial reports from TOKYO’s SFTP server is easily accomplished using the method that we already described in 9.2, “Distributing ad hoc travel package updates” on page 229.

In this section, we use **`fteAnt`** to distribute new packages to TOKYO. We use the same approach to distribute large software updates to TOKYO.

Instead of sending a file to TOKYO, we use the same method to *get* a file. Figure 9-23 shows a high-level view of how we accomplish using **fteAnt**.

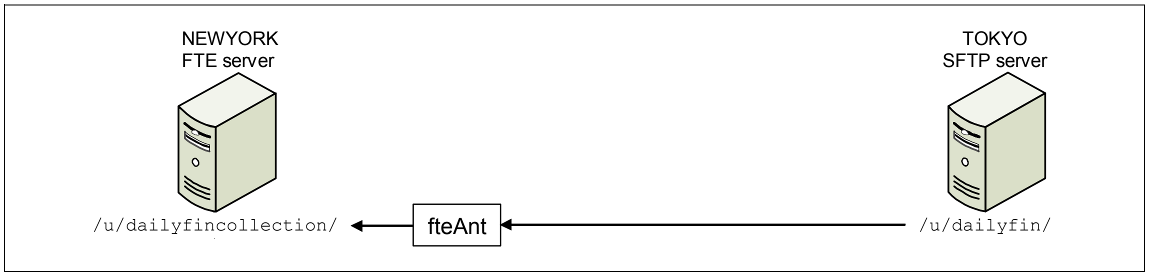


Figure 9-23 Using fteAnt to get a file

Setting up the transfer request

As a test of this method for retrieving daily financial reports from TOKYO, JKHL sends the daily report file, daily.financial.tokyo.report.txt, and a trigger file, daily.financial.tokyo.report.txt.trigge, to the /u/dailyfin/collection directory on its NEWYORK machine. The combined size of these daily report files is 10 to 50 K.

Example 9-19 shows the command to initiate, from WASHINGTON, the retrieval of the daily update files from the TOKYO machine.

Example 9-19 fteAnt command

```
fteAnt -f c:\dailyfin_tokyo.xml
```

Example 9-20 contains the specifications for the transfer that are contained in the dailyfin_tokyo.xml file.

Example 9-20 dailyfin_tokyo.xml

```
<?xml version='1.0'?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs">
<!-- From TOKYO machine -->
   <fte:call
      cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
        agent="NY.AGENT@NYQM"
      rcproperty="call.rc">
      <fte:command command="curl" successrc="0" retrycount="0" retrywait="5">
         <fte:arg value="-o"/>
         <fte:arg value="/u/dailyfincollection/daily.financial.TOKYO.report.txt" />
         <fte:arg value="--netrc"/>
         <fte:arg value="sftp://tokyo/u/dailyfin/daily.financial.TOKYO.report.txt"/>
      </fte:command>
   </fte:call>
   <fte:call
      cmdqm="NYQM@newyork@2415@SYSTEM.DEF.SVRCONN"
        agent="NY.AGENT@NYQM"
      rcproperty="call.rc">
      <fte:command command="curl" successrc="0" retrycount="0" retrywait="5">
         <fte:arg value="-o"/>
         <fte:arg 
value="/u/dailyfincollection/daily.financial.TOKYO.report.txt.trigger" />
         <fte:arg value="--netrc"/>
         <fte:arg 
value="sftp://tokyo/u/dailyfin/daily.financial.TOKYO.report.txt.trigger"/>
         <fte:arg value="--stderr" />
         <fte:arg value="curl.log" />
      </fte:command>
   </fte:call>
</project>
```

This successful XML command file can now be incorporated into the automated financial reporting system that we developed in Chapter 8, “Phase 2: Multi-step transfers” on page 191. In addition, the automated report collection steps that we developed in Chapter 8, “Phase 2: Multi-step transfers” on page 191, automatically feed into the back end database update, which we show in the next section.

9.3.2 Using WebSphere Message Broker as an interface to the JKHL CICS database update application

The last step of the existing FTE-based financial reporting system is to automatically send a daily consolidated financial report (and a trigger file) to the /u/etk0001/consolfinrpt.mb.in directory of the JKHL CHICAGO z/OS system, shown in Figure 9-24 on page 256. FTE sends these files with FTE’s code page conversion capability enabled; therefore, the files arrive in EBCDIC format.

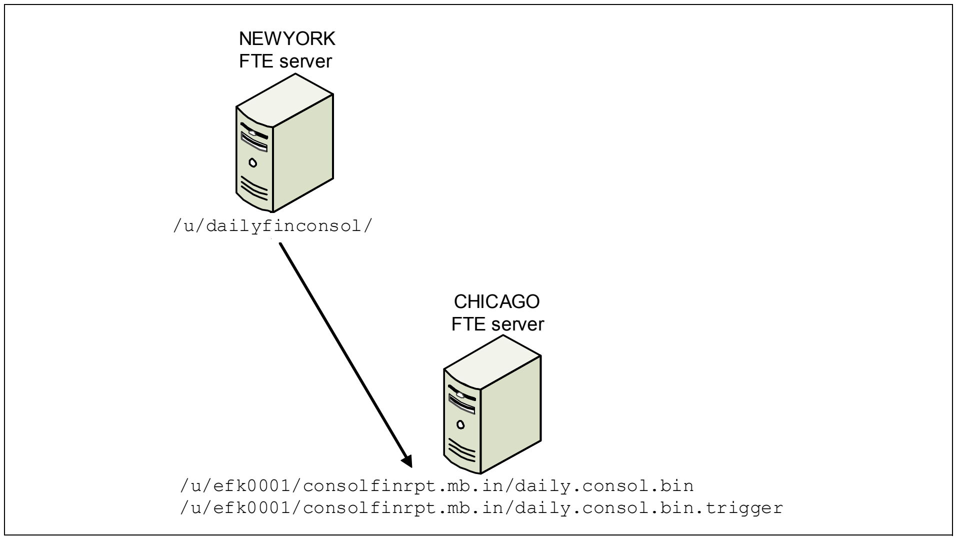

NEWYORK
FTE server
/u/dailyfinconsol/
CHICAGO
FTE server
/u/efk0001/consolfinrpt.mb.in/daily.consol.bin
/u/efk0001/consolfinrpt.mb.in/daily.consol.bin.trigger

Figure 9-24 Consolidated financial report

The two files arrive in a directory that is monitored by the WebSphere Message Broker's (Message Broker) fileInput node. When the trigger file arrives, the Message Broker:

1. Reads the daily.consol.bin file using a fileInput node.
2. Transforms the data into an existing COBOL copybook format.
3. Sends the financial report data to CICS, and then CICS updates the corporate database.

Figure 9-25 on page 257 shows a high-level view of how we accomplish our goal.

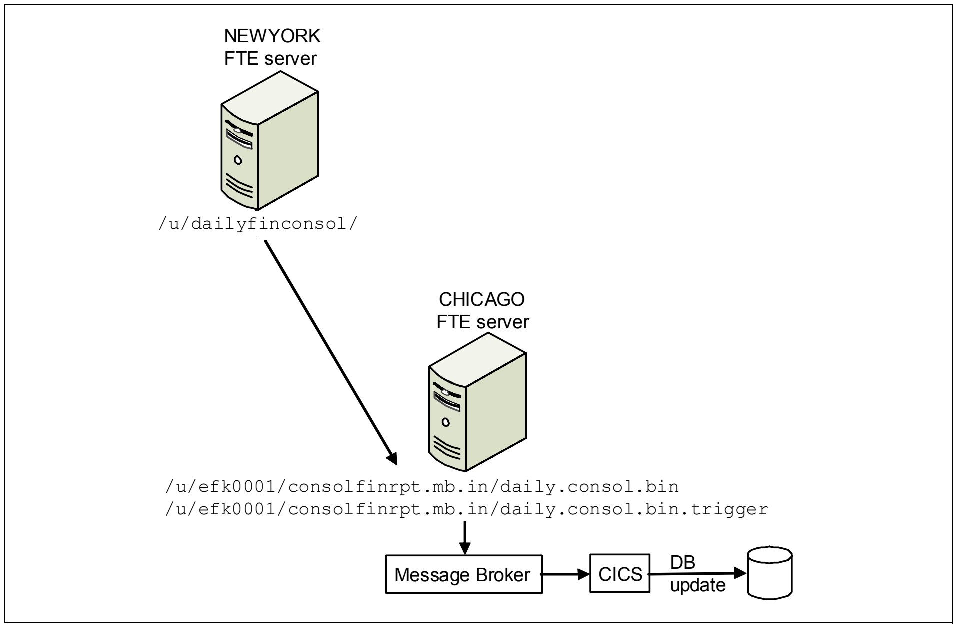

NEWYORK
FTE server
/u/dailyfinconsol/
CHICAGO
FTE server
/u/efk0001/consolfinrpt.mb.in/daily.consol.bin
/u/efk0001/consolfinrpt.mb.in/daily.consol.bin.trigger
Message Broker
CICS
DB
update

Figure 9-25 Message Broker integration

The financial reporting system is now complete.

9.3.3 Discussion of the results

The addition of WebSphere Message Broker to the JKHL FTE financial reporting system illustrates a simple method for using file-based integration with enterprise service bus software. We further discuss this topic in Chapter 14, “Integration” on page 349.

9.4 Receiving and distributing pricing updates

Occasionally, analysts at NY_FINANCIAL introduce pricing updates. These updates consist of a text file that has the current pricing for all of JKHL’s travel packages. Each update is usually 10 to 50 K.

9.4.1 The requirement

In the past, JKHL communicated pricing updates to its offices using a wide variety of manual ad hoc methods: e-mail, FTP, and mail. It was difficult to ensure that these critical price changes reached each office and were implemented in a timely manner. To make matters worse, none of these methods were secure.

Further, it is required that these pricing changes are recorded in the JKHL relational database in CHICAGO. The process of communicating these updates to CHICAGO used the same ad hoc methods: e-mail, FTP, and mail. Entering information into their CHICAGO database was a manual process.

JKHL plans to use FTE to completely automate the entire process. Because they have a license for WebSphere Message Broker, they plan to use that product to augment the received pricing information, and then automatically update the CHICAGO database.

JKHL plans to complete this automated system in these four steps:

1. NY_FINANCIAL sends raw pricing updates to CHICAGO.
2. WebSphere Message Broker processes the raw pricing update.
3. CHICAGO sends the final pricing update to NEWYORK for distribution.
4. NEWYORK distributes the final pricing update to all offices.

We present details about how we accomplished these steps in the following sections.

9.4.2 NY_FINANCIAL sends raw pricing update to CHICAGO

A financial analyst in the NY_FINANCIAL machine initiates JKHL's new pricing update process by using FTE to transfer a raw pricing update to CHICAGO.

Figure 9-26 on page 259 shows an overview of this first step in the pricing update process.

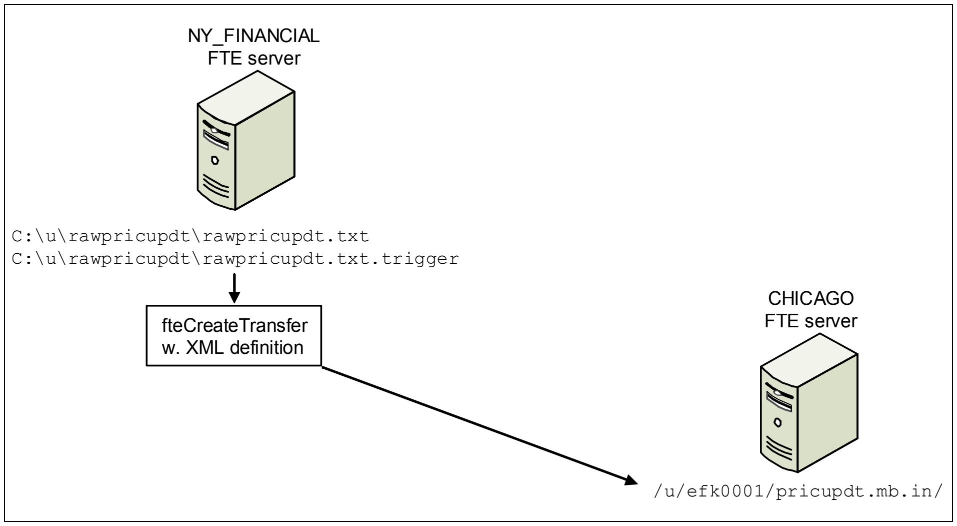

NY_FINANCIAL
FTE server
C:\u\rawpricupdt\rawpricupdt.txt
C:\u\rawpricupdt\rawpricupdt.txt.trigger
fteCreateTransfer
w. XML definition
CHICAGO
FTE server
/u/efk0001/pricupdt.mb.in/

Figure 9-26 Pricing update process

Setting up the transfer request

The JHKL analyst at NY_FINANCIAL sends the following files to CHICAGO:

- rawpricupdt.txt
- rawpricupdt.txt.trigger

Example 9-21 shows the **fteCreateTransfer** command that initiates the transfer to the CHICAGO machine.

Example 9-21 fteCreateTransfer

```
fteCreateTransfer
      -sa NYFIN.AGENT
      -sm NYQM
      -da CHICAGO.AGENT
      -dm MQH1
      -td nyfin_chicago.xml
```

Run the **fteCreateTransfer** command. The specifications for this transfer are contained in the nyfin_chicago.xml file, which we show in Example 9-22.

Example 9-22 myfin_chicago.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
```

```
  <item checksumMethod="none" mode="text">
    <source recursive="false" disposition="leave">
      <file encoding="UTF8"
EOL="CRLF">c:\u\rawpricupdt\rawpricupdt.txt</file>
    </source>
    <destination type="file" exist="overwrite">
      <file encoding="037"
EOL="CRLF">/u/efk0001/pricupdt.mb.in/rawpricupdt.txt</file>
    </destination>
  </item>
  <item checksumMethod="none" mode="text">
    <source recursive="false" disposition="leave">
      <file encoding="UTF8"
EOL="CRLF">c:\u\rawpricupdt\rawpricupdt.txt.trigger</file>
    </source>
    <destination type="file" exist="overwrite">
      <file encoding="037"
EOL="CRLF">/u/efk0001/pricupdt.mb.in/rawpricupdt.txt.trigger</file>
    </destination>
  </item>
</transferSpecifications>
```

Both files are converted from ASCII (UTF8) to EBCDIC (037). The trigger file is sent *after* the rawupdt.txt data file transfer is complete. In this manner, the arrival of the trigger file can be used to indicate that the *entire* data file arrived.

9.4.3 WebSphere Message Broker processes raw pricing update

The arrival at CHICAGO of the raw price update file and its associated trigger file causes WebSphere Message Broker's (Message Broker) fileInput node to read the raw price update. Message Broker then augments this price update with information from the JKHL relational database. Message Broker then updates the database and writes a final (augmented) pricing update file and an associated trigger file. Figure 9-27 on page 261 shows this process.

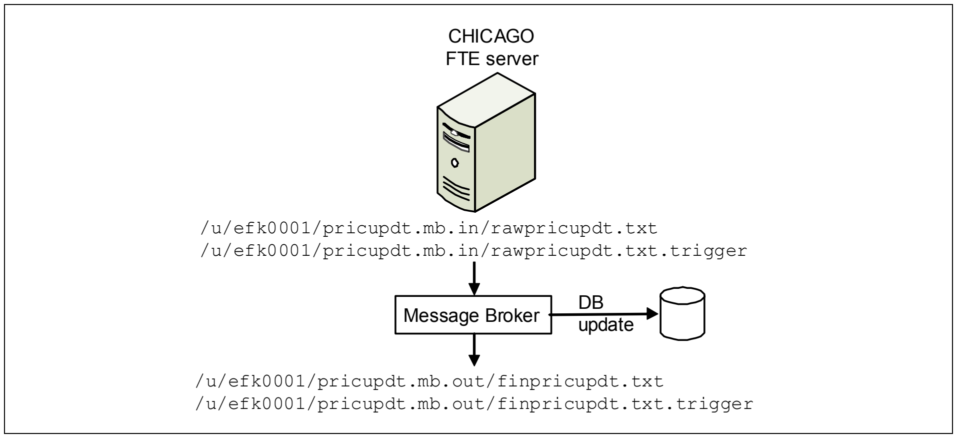

CHICAGO
FTE server
/u/efk0001/pricupdt.mb.in/rawpricupdt.txt
/u/efk0001/pricupdt.mb.in/rawpricupdt.txt.trigger
Message Broker
DB
update
/u/efk0001/pricupdt.mb.out/finpricupdt.txt
/u/efk0001/pricupdt.mb.out/finpricupdt.txt.trigger

Figure 9-27 Message Broker processes raw data

When the Message Broker writes these files, the next step in the pricing update process is triggered, which we describe in the next section.

9.4.4 CHICAGO sends final pricing updates to NEWYORK for distribution

At this stage of the pricing update process, the final pricing update that Message Broker creates is sent to NEWYORK for distribution.

We accomplish this step automatically when the final pricing update file and its associated trigger file arrive.

An FTE Resource Monitor with an associated XML command file is defined to watch the Message Broker's output directory for the arrival of a new final pricing update and trigger. It then sends these files to NEWYORK.

Figure 9-28 on page 262 shows this process.

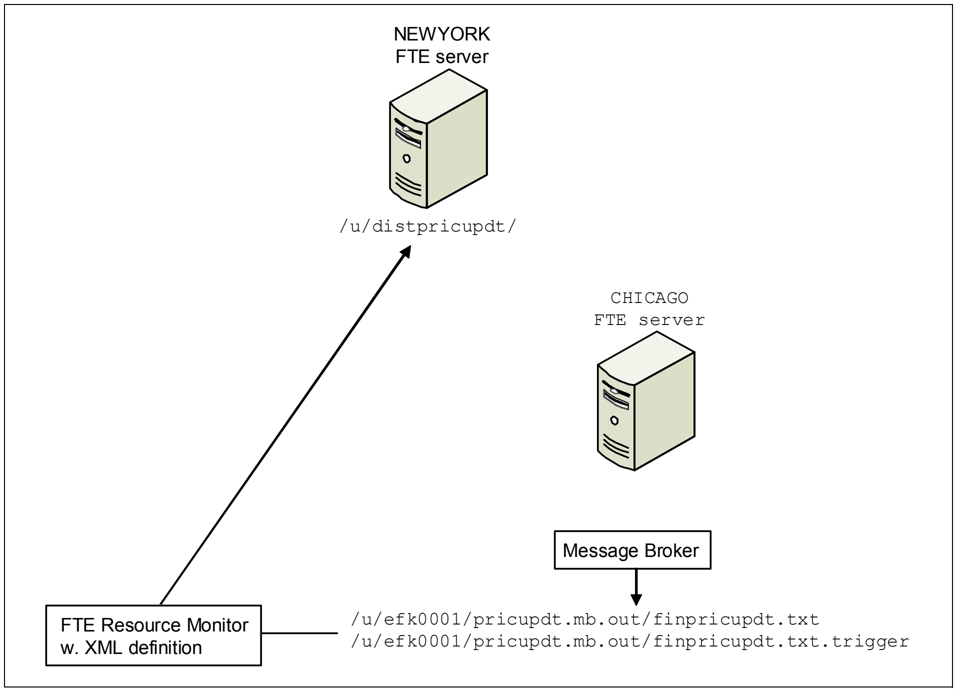


Figure 9-28 Final pricing update

Setting up the transfer request

A thorough discussion of how to set up the Resource Monitor is in Chapter 8, “Phase 2: Multi-step transfers” on page 191. In this section, we merely show the specific commands that were used to set up this particular monitor.

For setting up the automatic transfer, the **fteCreateMonitor** command is run on the WASHINGTON machine, as shown in Example 9-23.

Example 9-23 fteCreateMonitor command

```
fteCreateMonitor
      -ma CHICAGO.AGENT
      -mm MQH1
      -md /u/efk0001/pricupdt.mb.out
      -mn chicago_finpricupdt_monitor
      -mt c:\chicago_finpricupdt.xml
      -pi 10
      -pu seconds
      -tr match,finpricupdt.txt.trigger
```

The key things to notice in the **fteCreateMonitor** command are:

- The directory to be monitored: pricupdt.mb.out
- The file that triggers the file transfer process: finpricupdt.txt.trigger
- The file that contains the directions as to what to do when the trigger arrives: chicago_finpricupdt.xml

Example 9-24 shows the chicago_finpricupdt.xml command file.

Example 9-24 chicago_finpricupdt.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>washington.raleigh.ibm.com.</hostName>
        <userID>David</userID>
    </originator>
    <sourceAgent agent="CHICAGO.AGENT"
                 QMgr="MQH1"/>
    <destinationAgent agent="NY.AGENT"
                      QMgr="NYQM"/>
    <transferSet>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/u/efk0001/pricupdt.mb.out/finpricupdt.txt</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/distpricupdt/finpricupdt.txt</file>
        </destination>
      </item>
      <item mode="text" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/u/efk0001/pricupdt.mb.out/finpricupdt.txt.trigger</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/u/distpricupdt/finpricupdt.txt.trigger</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

This XML command file is run whenever Message Broker puts a new trigger file and final price update in the directory that this monitor is watching. The XML file directs FTE to send a request to the CHICAGO agent to send the two files to the NEWYORK agent where the final pricing update is distributed to all JKHL offices.

9.4.5 Discussion of the results

Using WebSphere MQ File Transfer Edition and WebSphere Message Broker, JKHL achieved a completely automated pricing update system.

This automation was accomplished using the following components:

- The FTE Resource Monitor to automatically move files to CHICAGO and then back to NY for distribution.
- WebSphere Message Broker:
  - Automatically read the raw price data
  - Enriched the data with information in a database
  - Updated a database
  - Transformed the data into a final price update that it wrote as a file where it was picked up by the FTE Resource Monitor

This scenario suggests the kind of integration solutions that can be achieved by combining managed file transfer with the Message Broker’s enterprise service bus capabilities.

Part 3

Additional use of WebSphere MQ File Transfer Edition

10 File transfer auditing and logging

In this chapter, we provide information about how to monitor file transfers that are in progress, and how to audit the file transfer activity using WebSphere MQ File Transfer Edition. Using FTE you can customize the audit and log information and integrate the transfer logs with an SQL database.

The topics that we discuss in this chapter are:

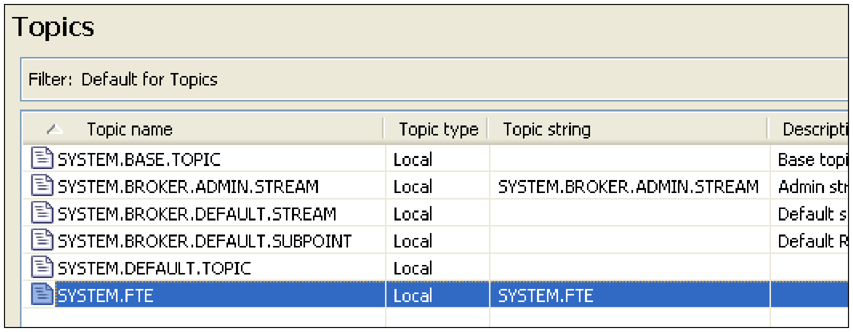
10.1 Overview

WebSphere MQ File Transfer Edition supports auditing transfer history and logging the transfer status. You can view the current transfer progress from the **Managed File Transfer** → **Current Transfer Progress** tab using the FTE plug-in of WebSphere MQ Explorer. You can also audit the transfer history information in detail from the Transfer Log window in WebSphere MQ Explorer. Although the WebSphere MQ Explorer cannot retain the transfer log and history records, the DatabaseLogger tool helps to solve the problem that history information can be persistently stored in a DB2® or Oracle® database.

A coordination queue manager is usually located in some central management and monitoring center. It collects information about:

- What FTE agents are defined
- The progress of active transfers
- Audit information about completed transfers

These messages are published to the SYSTEM.FTE queue in the coordination queue manager, and then the coordination queue manager publishes the messages on the queue to the SYSTEM.FTE topic, as shown in Figure 10-1.


Topics

Filter: Default for Topics

Topic name	Topic type	Topic string	Descripti
SYSTEM.BASE.TOPIC	Local		Base topi
SYSTEM.BROKER.ADMIN.STREAM	Local	SYSTEM.BROKER.ADMIN.STREAM	Admin str
SYSTEM.BROKER.DEFAULT.STREAM	Local		Default s
SYSTEM.BROKER.DEFAULT.SUBPOINT	Local		Default R
SYSTEM.DEFAULT.TOPIC	Local		
SYSTEM.FTE	Local	SYSTEM.FTE	

Figure 10-1 Displaying SYSTEM.FTE topic in WebSphere MQ Explorer

This coordination queue manager must be a WebSphere MQ V7 (or later) queue manager because WebSphere MQ File Transfer Edition relies on its publish-subscribe capabilities. But FTE is less fussy about the versions of the agent and command queue managers in the MQ network. Agents happily operate when connected to both version 6 and version 7 queue managers.

WebSphere MQ V7.0 provides a new Publish/Subscribe engine that is integrated into the queue manager, which is a major enhancement because the queue manager now internally manages all of the publish/subscribe functionality. The queue manager receives messages from publishers and subscription requests from subscribers for a range of topics and is responsible for queuing and routing these messages to the target subscribers. The WebSphere MQ V7.0 Information Center provides information about Publish/Subscribe, which expands on the information in this book. It is available at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.amqnar.doc/ps10120_.htm

You can see the SYSTEM.FTE topic tree on the coordination queue manager using WebSphere MQ Explorer, as shown in Figure 10-2. There are subtopics under the root of the SYSTEM.FTE topic: Agents, Log, Scheduler, Templates, Transfers, and so on.

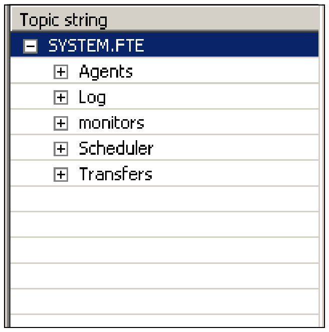
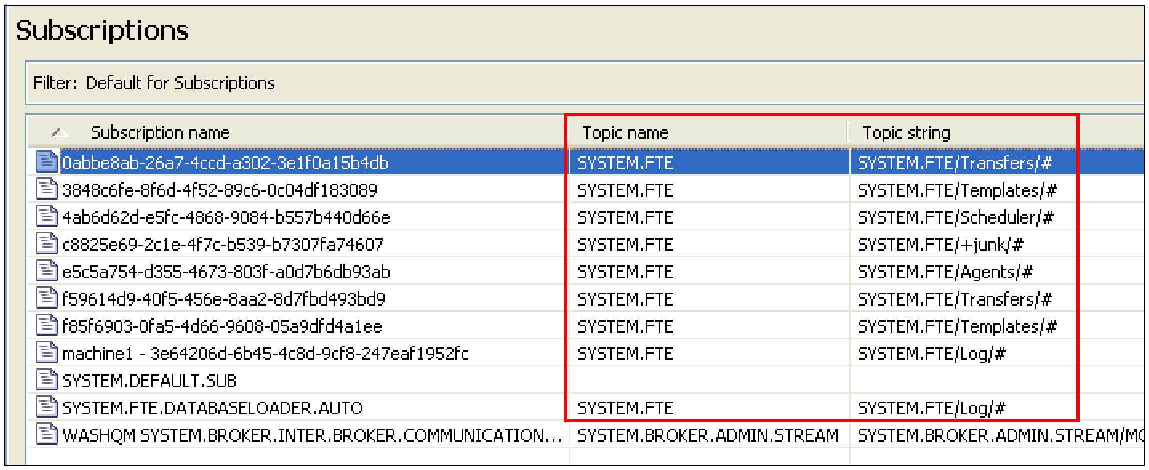


Figure 10-2   SYSTEM.FTE topic tree

At the start, end, and periodically during a file transfer, the source agent sends messages that contain audit-type information and messages that contain progress-type information to a coordination queue manager. The coordination queue manager publishes this information by different topics to its subscribers, which typically include instances of the WebSphere MQ Explorer plug-in. Other subscribers, such as third-party monitoring tools or tools for logging audit information to a database, can also subscribe to this data that they are interested in, as shown in Figure 10-3 on page 270.


Subscriptions

Filter: Default for Subscriptions

Subscription name	Topic name	Topic string
0abbe8ab-26a7-4ccd-a302-3e1f0a15b4db	SYSTEM.FTE	SYSTEM.FTE/Transfers/#
3848c6fe-8f6d-4f52-89c6-0c04df183089	SYSTEM.FTE	SYSTEM.FTE/Templates/#
4ab6d62d-e5fc-4868-9084-b557b440d66e	SYSTEM.FTE	SYSTEM.FTE/Scheduler/#
c8825e69-2c1e-4f7c-b539-b7307fa74607	SYSTEM.FTE	SYSTEM.FTE/+junk/#
e5c5a754-d355-4673-803f-a0d7b6db93ab	SYSTEM.FTE	SYSTEM.FTE/Agents/#
f59614d9-40f5-456e-8aa2-8d7fbd493bd9	SYSTEM.FTE	SYSTEM.FTE/Transfers/#
f85f6903-0fa5-4d66-9608-05a9dfd4a1ee	SYSTEM.FTE	SYSTEM.FTE/Templates/#
machine1 - 3e64206d-6b45-4c8d-9cf8-247eaf1952fc	SYSTEM.FTE	SYSTEM.FTE/Log/#
SYSTEM.DEFAULT.SUB		
SYSTEM.FTE.DATABASELOADER.AUTO	SYSTEM.FTE	SYSTEM.FTE/Log/#
WASHQM SYSTEM.BROKER.INTER.BROKER.COMMUNICATION...	SYSTEM.BROKER.ADMIN.STREAM	SYSTEM.BROKER.ADMIN.STREAM/MQ

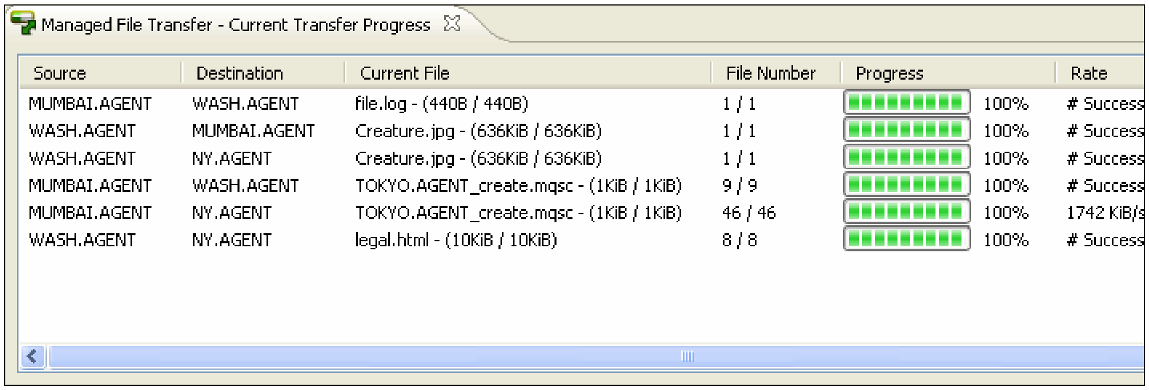
Figure 10-3 Subscribing the SYSTEM.FTE topics from the FTE Plug-in or other monitoring tools

10.2 Monitoring file transfers in process

You can monitor file transfers that are in progress using the Managed File Transfer - Current Transfer Progress tab in WebSphere MQ Explorer. These file transfers can be started using any of the following methods:

- WebSphere MQ Explorer
- Command line
- File transfer XML request message
- Scheduled or triggered file transfer
- Resource monitor
- `fteAnt` script request

After you start new file transfers using these methods, click the **Current Transfer Progress** tab to see the real-time progress of file transfers. You can see which file is sending and which file finished transferring, as shown in Figure 10-4 on page 271.



Managed File Transfer - Current Transfer Progress

Source	Destination	Current File	File Number	Progress		Rate
MUMBAI.AGENT	WASH.AGENT	file.log - (440B / 440B)	1 / 1		100%	# Success
WASH.AGENT	MUMBAI.AGENT	Creature.jpg - (636KiB / 636KiB)	1 / 1		100%	# Success
WASH.AGENT	NY.AGENT	Creature.jpg - (636KiB / 636KiB)	1 / 1		100%	# Success
MUMBAI.AGENT	WASH.AGENT	TOKYO.AGENT_create.mqsc - (1KiB / 1KiB)	9 / 9		100%	# Success
MUMBAI.AGENT	NY.AGENT	TOKYO.AGENT_create.mqsc - (1KiB / 1KiB)	46 / 46		100%	1742 KiB/s
WASH.AGENT	NY.AGENT	legal.html - (10KiB / 10KiB)	8 / 8		100%	# Success

Figure 10-4 Current Transfer Progress tab

The following information is displayed for each transfer in progress on the Current Transfer Progress tab:

- Source

  The name of the agent that sends the file from the source system.

- Destination

  The name of the agent that receives the file at the destination system.

- Current file

  The name of the file that is currently being transferred. The part of the individual file that was already transferred is displayed in B, KiB, MiB, GiB, or TiB along with total size of the file in parentheses. The unit of measurement that is 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.

- File number

  If you are transferring more than one file, this number represents how far through the total group of files the transfer is.

- Progress

  The progress bar shows how complete the current file transfer is as a percentage.

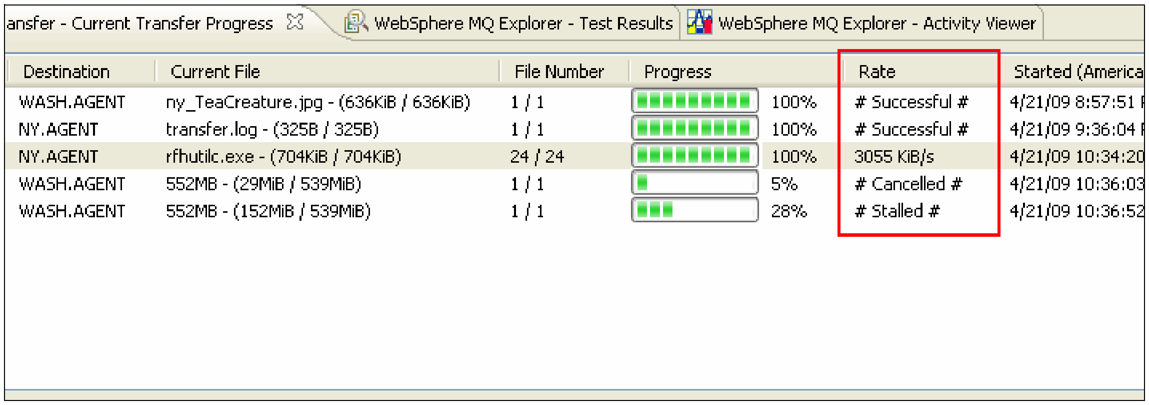
- Rate

  The rate the file is being transferred in KiB/s (kibibytes per second, where 1 kibibyte equals 1024 bytes). During the normal transfer, the value displays the transfer rate every second, as shown in Figure 10-5. If the file transfer is not in progress, the rate can display the different values:

  – When the transfer finishes, the value is Successful.

  – If the transfer is cancelled, the rate displays Cancelled.

  – When a file transfer stops temporarily for a while because of network or other available exception, the rate shows Stalled.


ansfer - Current Transfer Progress
WebSphere MQ Explorer - Test Results
WebSphere MQ Explorer - Activity Viewer

Destination	Current File	File Number	Progress	Rate	Started (America
WASH.AGENT	ny_TeaCreature.jpg - (636KiB / 636KiB)	1 / 1	100%	# Successful #	4/21/09 8:57:51
NY.AGENT	transfer.log - (325B / 325B)	1 / 1	100%	# Successful #	4/21/09 9:36:04
NY.AGENT	rfhutilc.exe - (704KiB / 704KiB)	24 / 24	100%	3055 KiB/s	4/21/09 10:34:20
WASH.AGENT	552MB - (29MiB / 539MiB)	1 / 1	5%	# Cancelled #	4/21/09 10:36:03
WASH.AGENT	552MB - (152MiB / 539MiB)	1 / 1	28%	# Stalled #	4/21/09 10:36:52

Figure 10-5 Displaying some different values in the Rate item

- Started (selected time zone)

  The time the transfer started in the administrator's selected time zone. To change the time zone that is displayed:

  a. Click **Windows** → **Preferences** → **WebSphere MQ File Transfer Edition**, as shown in Figure 10-6 on page 273.

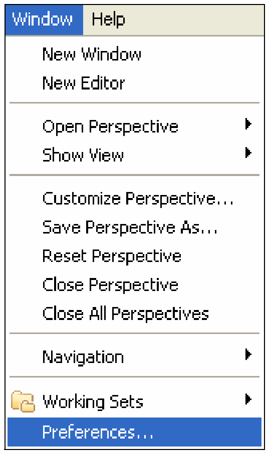
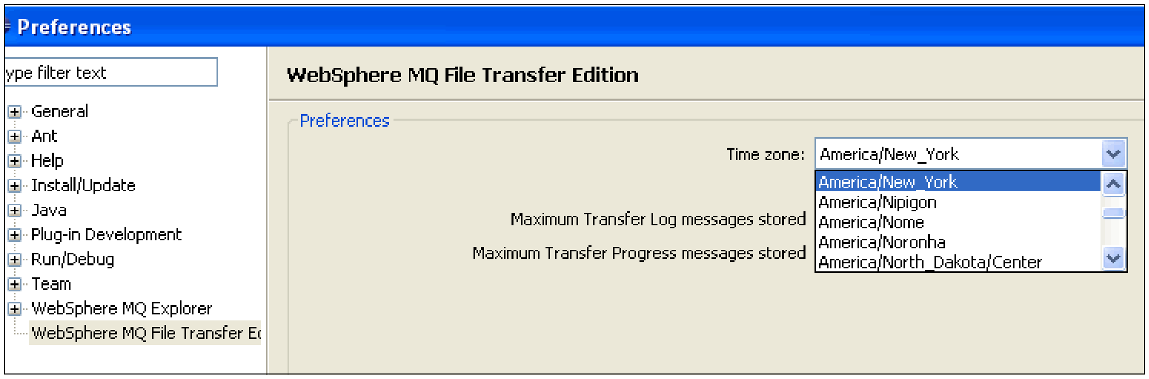


Figure 10-6 Select Preferences from the Window menu of WebSphere MQ Explorer

b. From the Time zone list, select an alternative time zone, as shown in Figure 10-7. Click **OK**



Time zone: America/New_York

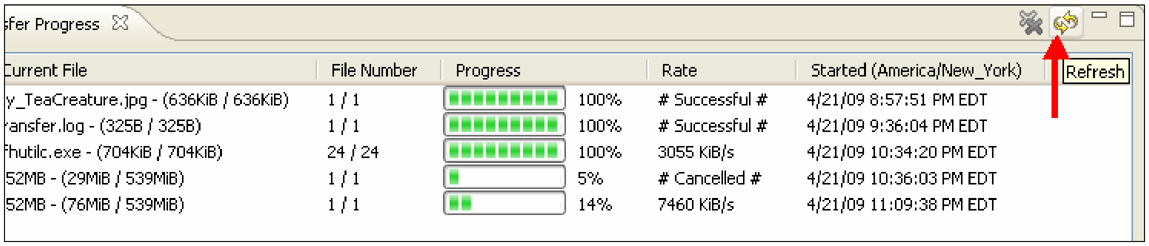
Maximum Transfer Log messages stored

Maximum Transfer Progress messages stored

America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Center

Figure 10-7 Time zone selection of file transfers in WebSphere MQ File Transfer Edition

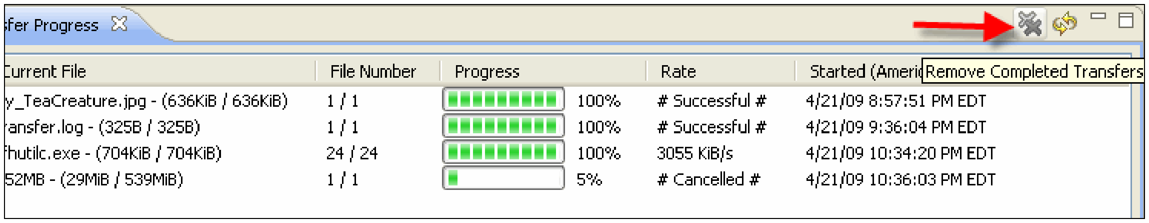
The Current Transfer Progress tab regularly refreshes its information automatically, but to force a refreshed view of what is displayed in the Current Transfer Progress tab, on the Content view toolbar click **Refresh**, Figure 10-8 on page 274. Alternately, you can press **F5**.



urrent File	File Number	Progress	Rate	Started (America/New_York)
y_TeaCreature.jpg - (636KiB / 636KiB)	1 / 1	100%	# Successful #	4/21/09 8:57:51 PM EDT
ansfer.log - (325B / 325B)	1 / 1	100%	# Successful #	4/21/09 9:36:04 PM EDT
hutilc.exe - (704KiB / 704KiB)	24 / 24	100%	3055 KiB/s	4/21/09 10:34:20 PM EDT
52MB - (29MiB / 539MiB)	1 / 1	5%	# Cancelled #	4/21/09 10:36:03 PM EDT
52MB - (76MiB / 539MiB)	1 / 1	14%	7460 KiB/s	4/21/09 11:09:38 PM EDT

Figure 10-8 Refreshing the transfer status in progress

If you want to delete file transfers from the Current Transfer Progress tab, on the Content view toolbar, click **Remove completed transfers**, as shown in Figure 10-9. Clicking this button removes file transfer details from the tab only. It does not stop or cancel a current or scheduled transfer.



urrent File	File Number	Progress	Rate	Started (Ameri
y_TeaCreature.jpg - (636KiB / 636KiB)	1 / 1	100%	# Successful #	4/21/09 8:57:51 PM EDT
ansfer.log - (325B / 325B)	1 / 1	100%	# Successful #	4/21/09 9:36:04 PM EDT
hutilc.exe - (704KiB / 704KiB)	24 / 24	100%	3055 KiB/s	4/21/09 10:34:20 PM EDT
52MB - (29MiB / 539MiB)	1 / 1	5%	# Cancelled #	4/21/09 10:36:03 PM EDT

Figure 10-9 Removing the completed transfers

The Current Transfer Progress tab in the WebSphere MQ Explorer cannot persistently save any audit or status information to databases or monitor log files. Previous file transfer information is not retained after you stop and restart the WebSphere MQ Explorer. At restart, the information about past transfers is cleared from the Current Transfer Progress tab. In 10.5, “Archiving audit information in the database” on page 289, we introduce how to store transfer log and information in the database using the Database Loading command.

10.3 Viewing the status and history of file transfers

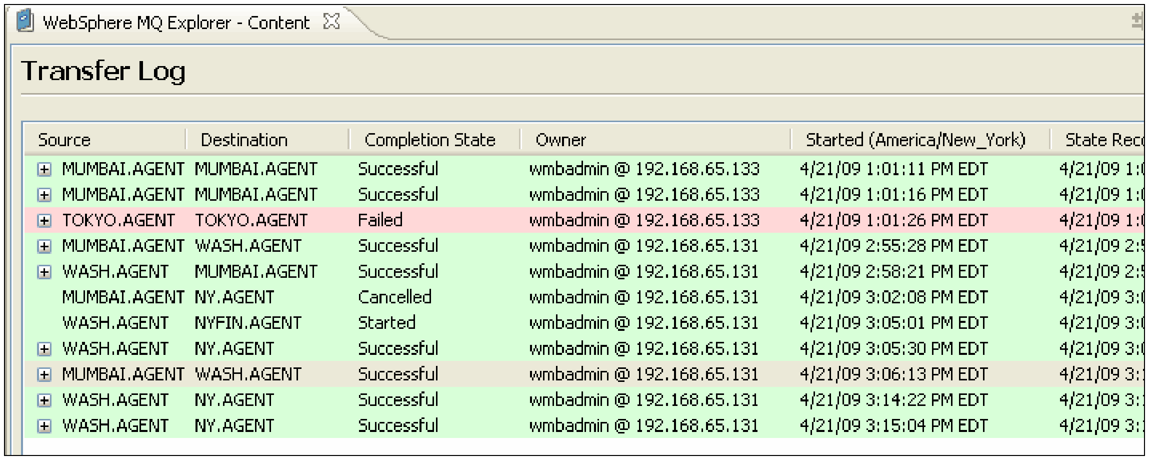
You can see the detail of file transfer status and history using the Transfer Log view in WebSphere MQ Explorer. These file transfer statuses can be transfers that are started using:

- WebSphere MQ Explorer
- Command line
- File transfer XML request message
- Scheduled or triggered file transfer

- Resource monitor
- fteAnt script

10.3.1 Viewing the transfer information in the Transfer Log

After you start a file transfer using any of these methods, when the file transfer completes, the Transfer Log window can display the transfer status and history. You can see the information of every submitted transfer request during the transfer or after the transfer completes, as shown in Figure 10-10.



WebSphere MQ Explorer - Content

Transfer Log

Source	Destination	Completion State	Owner	Started (America/New_York)	State Rec
MUMBAI.AGENT	MUMBAI.AGENT	Successful	wmbadmin @ 192.168.65.133	4/21/09 1:01:11 PM EDT	4/21/09 1:
MUMBAI.AGENT	MUMBAI.AGENT	Successful	wmbadmin @ 192.168.65.133	4/21/09 1:01:16 PM EDT	4/21/09 1:
TOKYO.AGENT	TOKYO.AGENT	Failed	wmbadmin @ 192.168.65.133	4/21/09 1:01:26 PM EDT	4/21/09 1:
MUMBAI.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 2:55:28 PM EDT	4/21/09 2:
WASH.AGENT	MUMBAI.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 2:58:21 PM EDT	4/21/09 2:
MUMBAI.AGENT	NY.AGENT	Cancelled	wmbadmin @ 192.168.65.131	4/21/09 3:02:08 PM EDT	4/21/09 3:
WASH.AGENT	NYFIN.AGENT	Started	wmbadmin @ 192.168.65.131	4/21/09 3:05:01 PM EDT	4/21/09 3:
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 3:05:30 PM EDT	4/21/09 3:
MUMBAI.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 3:06:13 PM EDT	4/21/09 3:
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 3:14:22 PM EDT	4/21/09 3:
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 3:15:04 PM EDT	4/21/09 3:

Figure 10-10 Displaying the transfer logs

To view the transfer log:

1. In the Navigator view, expand **Managed File Transfer**, and then expand the name of the coordination queue manager (for example WASHQM) that you want to view the transfer log for, as shown in Figure 10-11 on page 276.

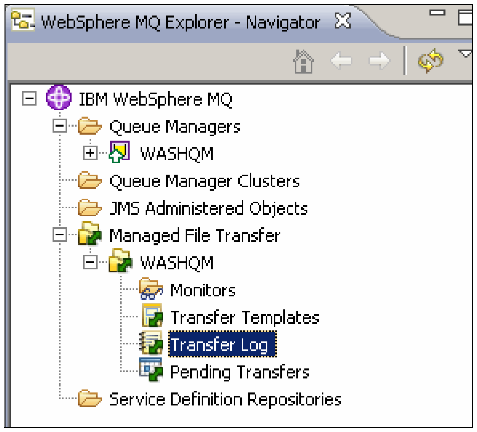


Figure 10-11 Opening the transfer log in Managed File Transfer

2. In the Navigator view, click **Transfer Log**. The Transfer Log is displayed in the Content view on the right.

The Transfer Log window displays the following information about file transfers:

- Source

  The name of the agent on the system where the source file is located.

- Destination

  The name of the agent on the system to which you want to transfer the file.

- Completion State

  The current status of the file transfer. The state can be one of the following values: Started, In progress, Successful, Partially Successful, Cancelled, or Failed.

- Owner

  The user ID on a particular host that owns the transferred file.

- Started (selected time zone)

  The time and date that the file transfer started in the administrator's selected time zone. To change the time zone displayed:

  a. Click **Window** → **Preferences** → **WebSphere MQ File Transfer Edition**.

  b. Select an alternative time zone from the Time zone list. Click **OK**.

- State Recorded (selected time zone)

  The time and date that the completion state was recorded, in the administrator's selected time zone. This column is not displayed by default. You can display the column using the Configure Transfer Log Columns window.

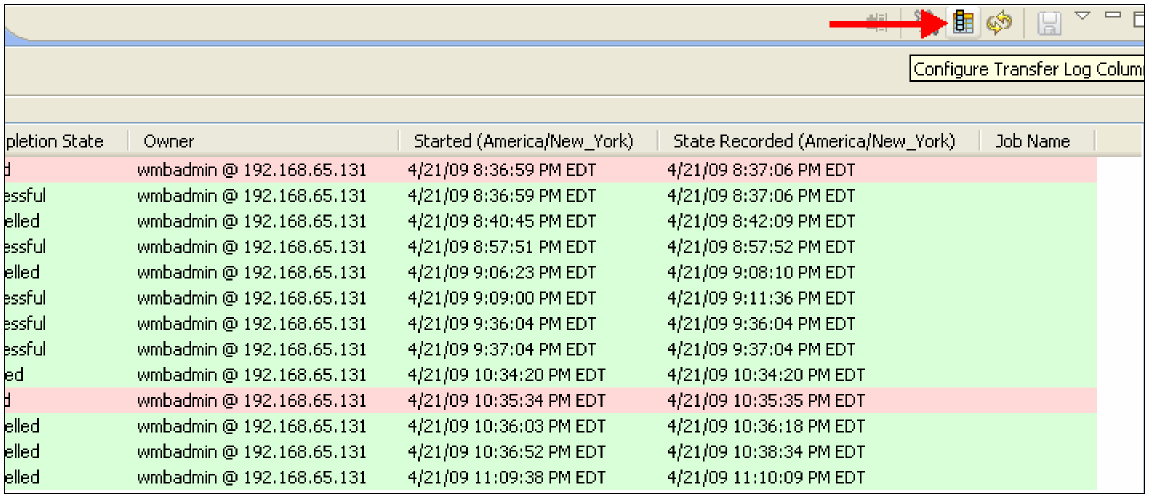
- Job Name

  Job name reference. A user-defined identifier for the entire transfer when using the **`fteCreateTransfer`** command with -jn parameter. This column is not displayed by default. You can display the column using the Configure Transfer Log Columns window.

10.3.2 Rearranging and displaying columns in the Transfer Log

You can configure the information that is displayed in the Transfer Log of WebSphere MQ Explorer. To customize which columns are displayed in the **Transfer Log**, use Configure Transfer Log Columns, and:

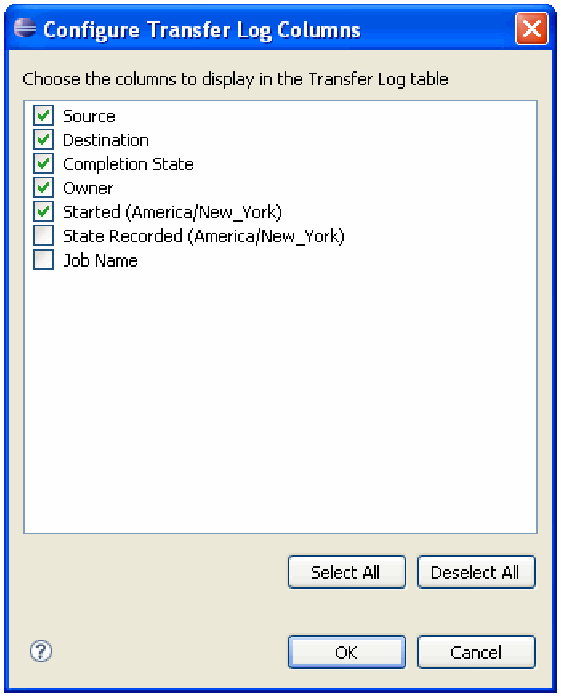
1. Open the Transfer Log to the Content view, and click **Configure Transfer Log Columns** on the Content view tool bar, as shown in Figure 10-12. The Configure Transfer Log Columns window opens.



pletion State	Owner	Started (America/New_York)	State Recorded (America/New_York)	Job Name
d	wmbadmin @ 192.168.65.131	4/21/09 8:36:59 PM EDT	4/21/09 8:37:06 PM EDT	
essful	wmbadmin @ 192.168.65.131	4/21/09 8:36:59 PM EDT	4/21/09 8:37:06 PM EDT	
elled	wmbadmin @ 192.168.65.131	4/21/09 8:40:45 PM EDT	4/21/09 8:42:09 PM EDT	
essful	wmbadmin @ 192.168.65.131	4/21/09 8:57:51 PM EDT	4/21/09 8:57:52 PM EDT	
elled	wmbadmin @ 192.168.65.131	4/21/09 9:06:23 PM EDT	4/21/09 9:08:10 PM EDT	
essful	wmbadmin @ 192.168.65.131	4/21/09 9:09:00 PM EDT	4/21/09 9:11:36 PM EDT	
essful	wmbadmin @ 192.168.65.131	4/21/09 9:36:04 PM EDT	4/21/09 9:36:04 PM EDT	
essful	wmbadmin @ 192.168.65.131	4/21/09 9:37:04 PM EDT	4/21/09 9:37:04 PM EDT	
ed	wmbadmin @ 192.168.65.131	4/21/09 10:34:20 PM EDT	4/21/09 10:34:20 PM EDT	
d	wmbadmin @ 192.168.65.131	4/21/09 10:35:34 PM EDT	4/21/09 10:35:35 PM EDT	
elled	wmbadmin @ 192.168.65.131	4/21/09 10:36:03 PM EDT	4/21/09 10:36:18 PM EDT	
elled	wmbadmin @ 192.168.65.131	4/21/09 10:36:52 PM EDT	4/21/09 10:38:34 PM EDT	
elled	wmbadmin @ 192.168.65.131	4/21/09 11:09:38 PM EDT	4/21/09 11:10:09 PM EDT	

Figure 10-12 The button of Configure Transfer Log Columns

2. To customize your view of the transfer log, select or deselect individual check boxes for the columns that you want to show or hide, as shown in Figure 10-13 on page 278, for example, you can select **State Recorded** and **Job Name** that are not displayed by default, and click **OK**.

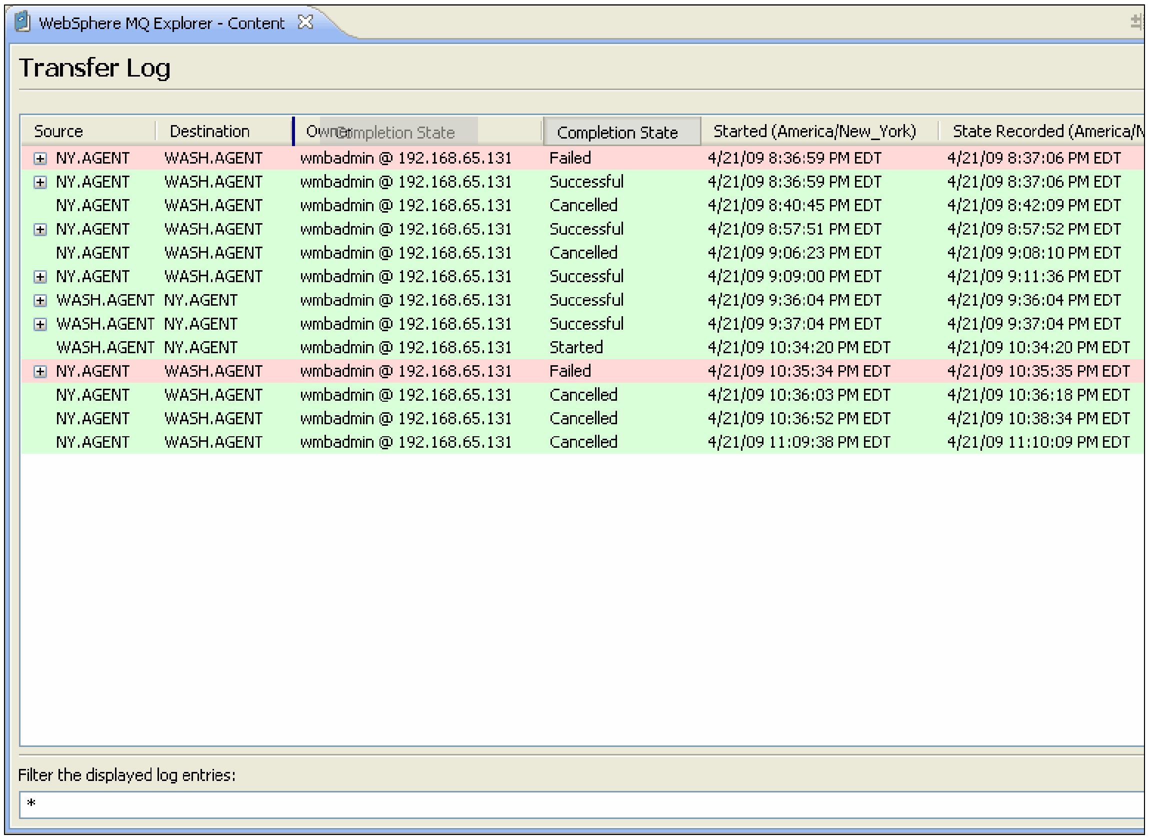


Choose the columns to display in the Transfer Log table

- [x] Source
- [x] Destination
- [x] Completion State
- [x] Owner
- [x] Started (America/New_York)
- [ ] State Recorded (America/New_York)
- [ ] Job Name

Figure 10-13 Configure Transfer Log Columns window

3. To rearrange the order of the columns in the Transfer Log, click the title of the column that you want to move and drag the column to its new position, as shown in Figure 10-14 on page 279. The new column order is retained until your next stop and restart of the WebSphere MQ Explorer.



WebSphere MQ Explorer - Content

Transfer Log

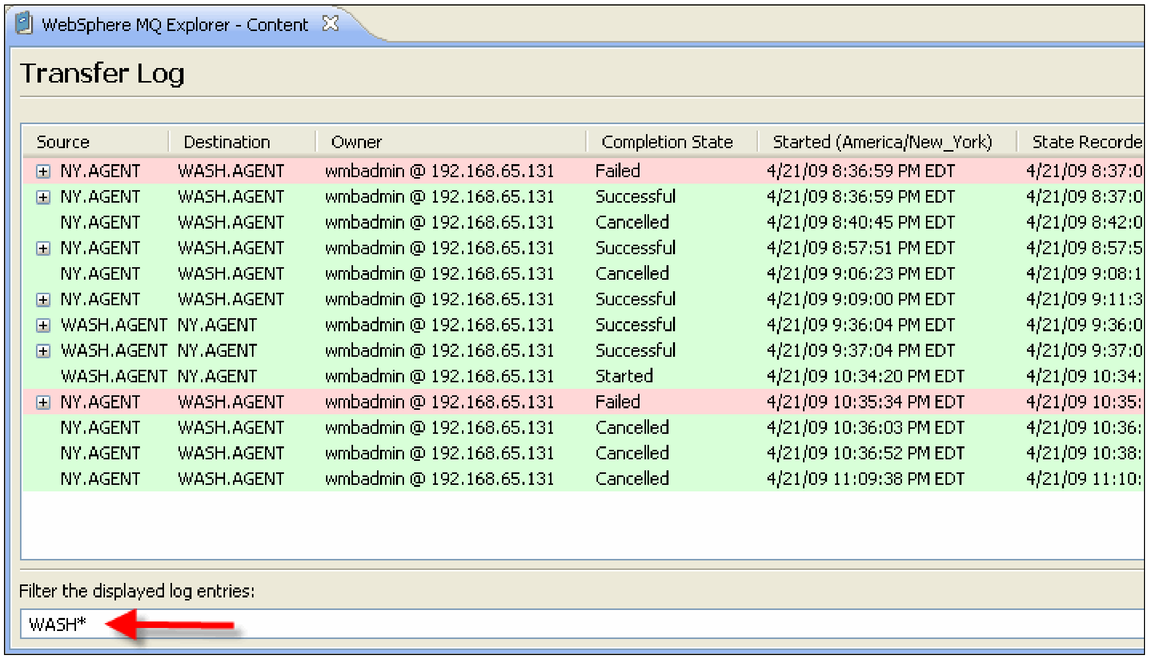
Source	Destination	Owner	Completion State	Started (America/New_York)	State Recorded (America/N
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Failed	4/21/09 8:36:59 PM EDT	4/21/09 8:37:06 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 8:36:59 PM EDT	4/21/09 8:37:06 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 8:40:45 PM EDT	4/21/09 8:42:09 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 8:57:51 PM EDT	4/21/09 8:57:52 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 9:06:23 PM EDT	4/21/09 9:08:10 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:09:00 PM EDT	4/21/09 9:11:36 PM EDT
WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:36:04 PM EDT	4/21/09 9:36:04 PM EDT
WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:37:04 PM EDT	4/21/09 9:37:04 PM EDT
WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Started	4/21/09 10:34:20 PM EDT	4/21/09 10:34:20 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Failed	4/21/09 10:35:34 PM EDT	4/21/09 10:35:35 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 10:36:03 PM EDT	4/21/09 10:36:18 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 10:36:52 PM EDT	4/21/09 10:38:34 PM EDT
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 11:09:38 PM EDT	4/21/09 11:10:09 PM EDT

Filter the displayed log entries:

*

Figure 10-14 Rearranging the columns on the Transfer Log

4. You can filter the transfer status and history list that are displayed in the Transfer Log. Enter a string in the Filter the displayed log entries field. Use the asterisk (*) character as a wildcard to represent zero or more characters, for example, you can set WASH* or Success* characters as the filter information. See Figure 10-15 on page 280.

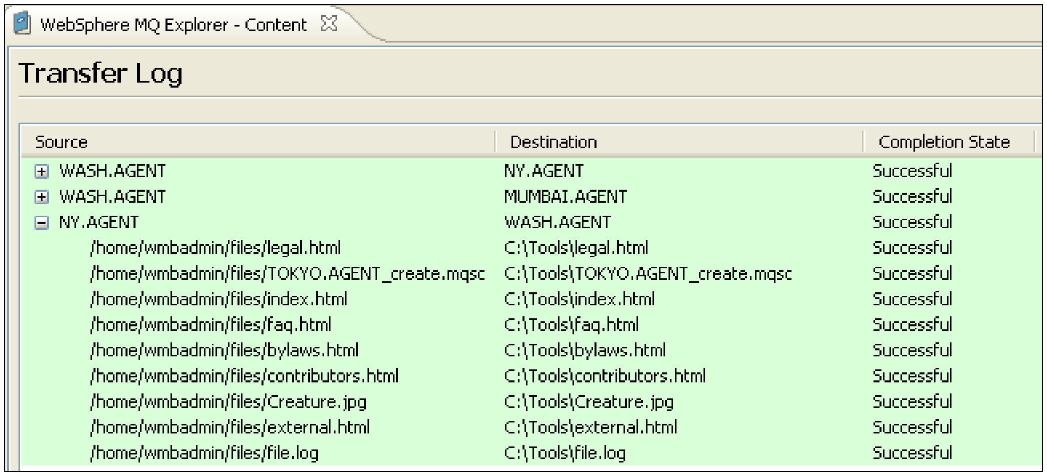


Source	Destination	Owner	Completion State	Started (America/New_York)	State Recorde
⊞ NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Failed	4/21/09 8:36:59 PM EDT	4/21/09 8:37:0
⊞ NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 8:36:59 PM EDT	4/21/09 8:37:0
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 8:40:45 PM EDT	4/21/09 8:42:0
⊞ NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 8:57:51 PM EDT	4/21/09 8:57:5
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 9:06:23 PM EDT	4/21/09 9:08:1
⊞ NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:09:00 PM EDT	4/21/09 9:11:3
⊞ WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:36:04 PM EDT	4/21/09 9:36:0
⊞ WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Successful	4/21/09 9:37:04 PM EDT	4/21/09 9:37:0
WASH.AGENT	NY.AGENT	wmbadmin @ 192.168.65.131	Started	4/21/09 10:34:20 PM EDT	4/21/09 10:34:
⊞ NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Failed	4/21/09 10:35:34 PM EDT	4/21/09 10:35:
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 10:36:03 PM EDT	4/21/09 10:36:
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 10:36:52 PM EDT	4/21/09 10:38:
NY.AGENT	WASH.AGENT	wmbadmin @ 192.168.65.131	Cancelled	4/21/09 11:09:38 PM EDT	4/21/09 11:10:

Filter the displayed log entries: WASH*

Figure 10-15 The filter of the transfer log

5. If you want to view further details about a completed transfer, click the plus sign(+), and then expand the transfer record. You can see all of the source and destination file names that are included in that transfer, as shown in Figure 10-16.

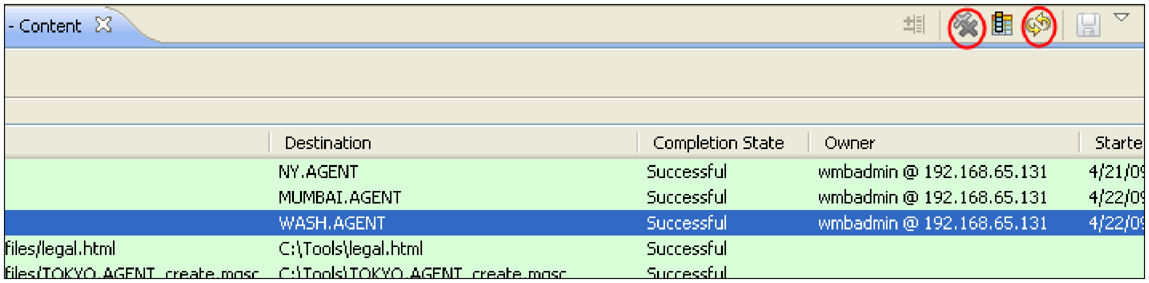


Source	Destination	Completion State
⊞ WASH.AGENT	NY.AGENT	Successful
⊞ WASH.AGENT	MUMBAI.AGENT	Successful
⊟ NY.AGENT	WASH.AGENT	Successful
/home/wmbadmin/files/legal.html	C:\Tools\legal.html	Successful
/home/wmbadmin/files/TOKYO.AGENT_create.mqsc	C:\Tools\TOKYO.AGENT_create.mqsc	Successful
/home/wmbadmin/files/index.html	C:\Tools\index.html	Successful
/home/wmbadmin/files/faq.html	C:\Tools\faq.html	Successful
/home/wmbadmin/files/bylaws.html	C:\Tools\bylaws.html	Successful
/home/wmbadmin/files/contributors.html	C:\Tools\contributors.html	Successful
/home/wmbadmin/files/Creature.jpg	C:\Tools\Creature.jpg	Successful
/home/wmbadmin/files/external.html	C:\Tools\external.html	Successful
/home/wmbadmin/files/file.log	C:\Tools\file.log	Successful

Figure 10-16 Transfer the file list in detail

Note: If the transfer is currently in progress and consists of a large number of files, you can only view the files that have were already transferred so far.

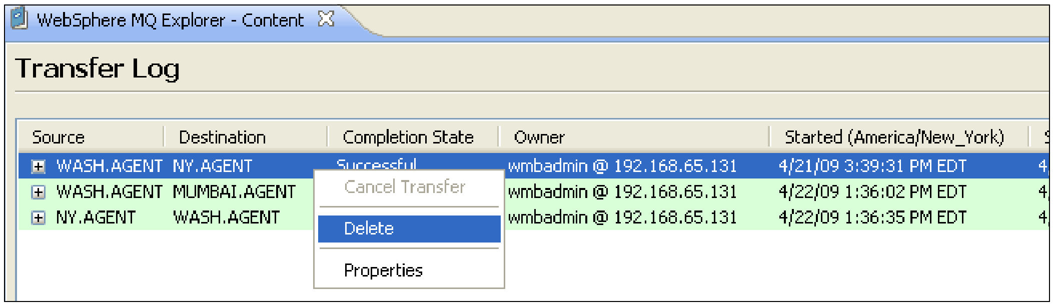
6. To refresh the Transfer Log display, on the Transfer Log view toolbar, click **Refresh**. The file transfer information in the Transfer Log remains in the log after you stop and restart the WebSphere MQ Explorer. If you want to delete all completed file transfers from the log, on the Content view toolbar, click **Remove Completed Transfers**, as shown in Figure 10-17.



	Destination	Completion State	Owner	Starte
	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/0
	MUMBAI.AGENT	Successful	wmbadmin @ 192.168.65.131	4/22/0
	WASH.AGENT	Successful	wmbadmin @ 192.168.65.131	4/22/0
files/legal.html	C:\Tools\legal.html	Successful		
files/TOKYO.AGENT_create.mqsc	C:\Tools\TOKYO.AGENT_create.mqsc	Successful		

Figure 10-17 Refresh and Remove buttons

7. To delete an individual completed file transfer from the transfer log, right-click the transfer, and click **Delete**, as shown in Figure 10-18.



Transfer Log

Source	Destination	Completion State	Owner	Started (America/New_York)	S
WASH.AGENT	NY.AGENT	Successful	wmbadmin @ 192.168.65.131	4/21/09 3:39:31 PM EDT	4
WASH.AGENT	MUMBAI.AGENT		wmbadmin @ 192.168.65.131	4/22/09 1:36:02 PM EDT	4
NY.AGENT	WASH.AGENT		wmbadmin @ 192.168.65.131	4/22/09 1:36:35 PM EDT	4

Figure 10-18 Deleting the transfer log

If a file transfer log is deleted, it does not stop or cancel a transfer that is in progress or that was scheduled. You only deleted the stored historical data.

10.3.3 Viewing history information in detail

You can see more transfer history details by viewing the properties of a transfer log entry in XML format. The history information describes all of the transfer attributes at start point, during the transfer process, and at the completion point. Example 10-1 shows an example of history information.

Example 10-1 Start log information as XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction ID="414d512057415348514d202020202020204e22ef4920005702"
   agentRole="sourceAgent" version="1.00"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="TransferLog.xsd">
   <action time="2009-04-22T17:36:35.648Z">started</action>
   <sourceAgent QMgr="NYQM" agent="NY.AGENT">
      <systemInfo architecture="x86" name="Linux" version="2.6.18-92.el5"/>
   </sourceAgent>
   <destinationAgent QMgr="WASHQM" agent="WASH.AGENT"/>
   <originator>
      <hostName>192.168.65.131</hostName>
      <userID>wmbadmin</userID>
      <mqmdUserID>musr_mqadmin</mqmdUserID>
   </originator>
   <transferSet startTime="2009-04-22T17:36:35.649Z" total="9">
      <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">NY.AGENT</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">WASH.AGENT</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">192.168.65.131</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">musr_mqadmin</metaData>
      <metaData
         key="com.ibm.wmqfte.TransferId">414d512057415348514d202020202020204e
               22ef4920005702</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">wmbadmin</metaData>
      </metaDataSet>
   </transferSet>
</transaction>
```

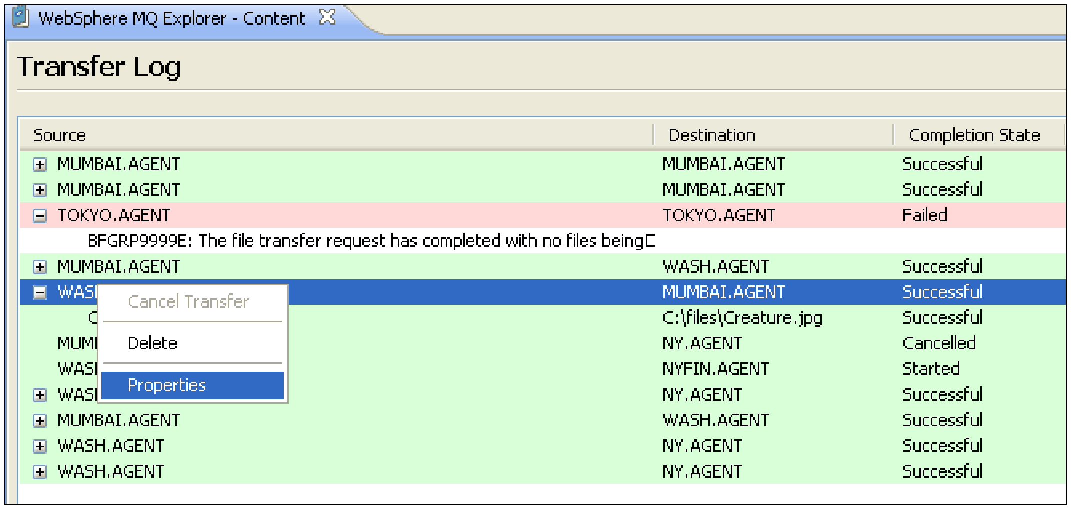
As shown in Example 10-1, the XML format message displays some attributes at the starting point. The *File transfer log message formats* section introduces all of the tags and attributes, which is located in the FTE Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/audit_message_format.htm

You can view history information from the Transfer Log view in the WebSphere MQ Explorer.

To view history information in the WebSphere MQ Explorer Transfer Log:

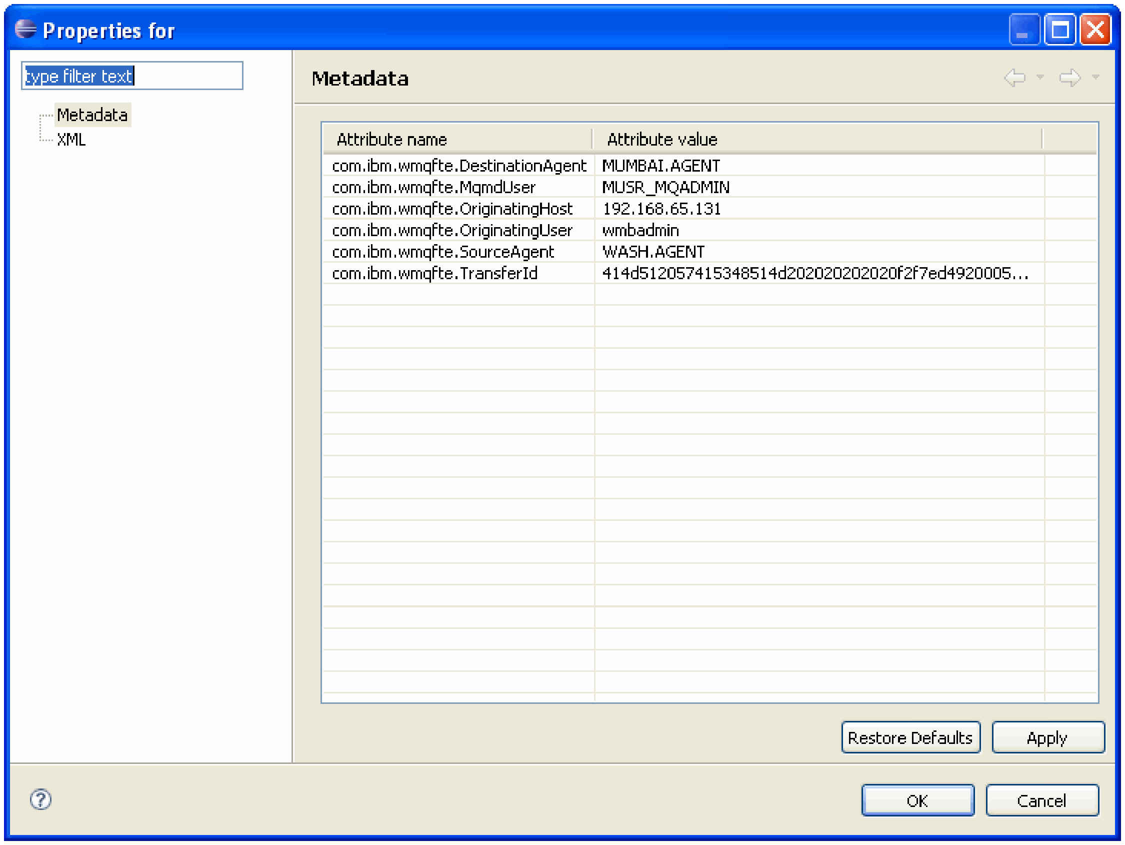
1. In the **Transfer Log** view, Select and right-click the transfer log item that you are interested in.
2. Select **Properties**, as show in Figure 10-19.



Source	Destination	Completion State
MUMBAI.AGENT	MUMBAI.AGENT	Successful
MUMBAI.AGENT	MUMBAI.AGENT	Successful
TOKYO.AGENT	TOKYO.AGENT	Failed
BFGRP9999E: The file transfer request has completed with no files being□		
MUMBAI.AGENT	WASH.AGENT	Successful
WASI	MUMBAI.AGENT	Successful
C	C:\files\Creature.jpg	Successful
MUMI	NY.AGENT	Cancelled
WASI	NYFIN.AGENT	Started
WASI	NY.AGENT	Successful
MUMBAI.AGENT	WASH.AGENT	Successful
WASH.AGENT	NY.AGENT	Successful
WASH.AGENT	NY.AGENT	Successful

Figure 10-19 Opening the transfer log property

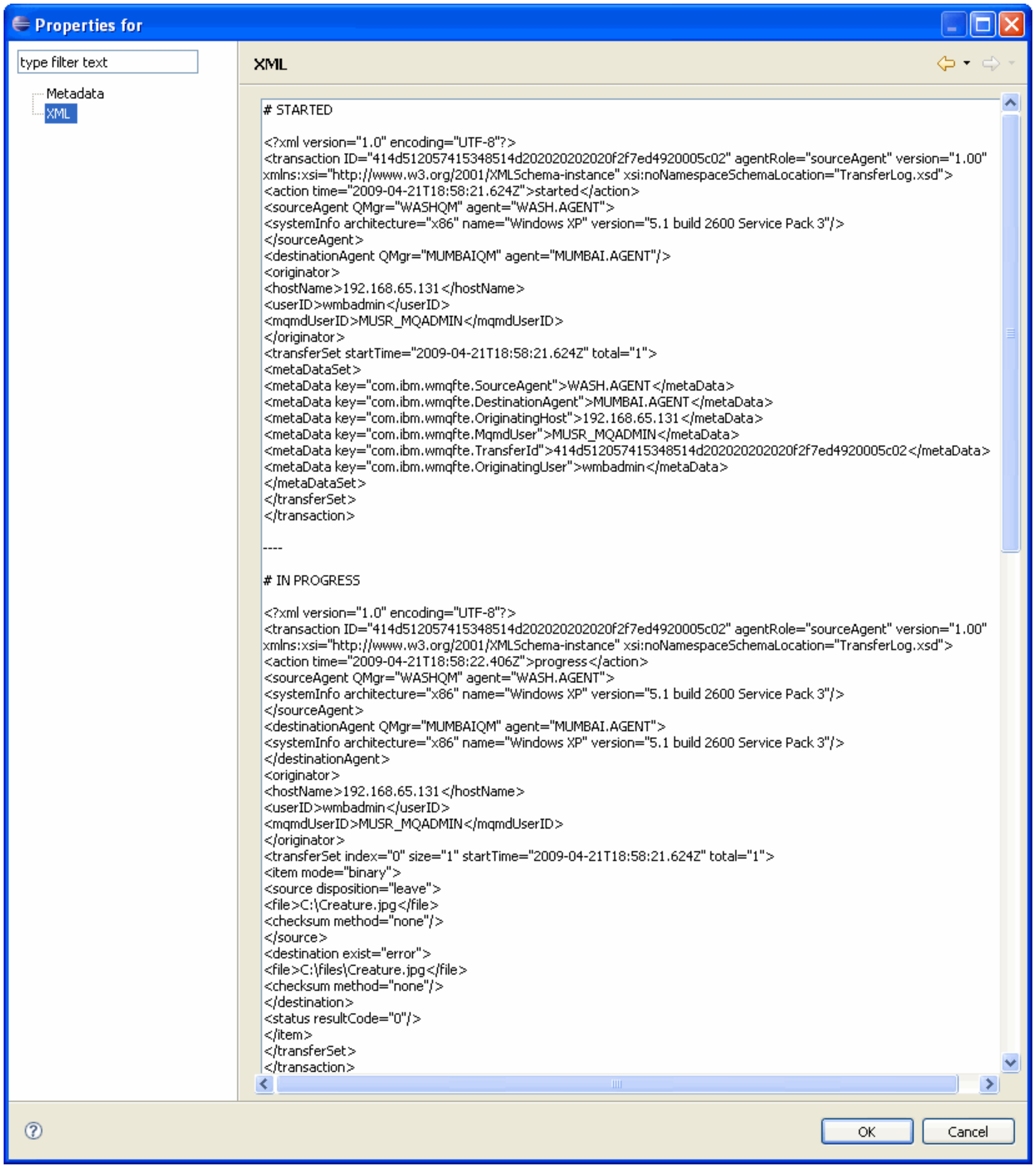
In the Metadata view, basic attributes are displayed, as shown in Figure 10-20 on page 284.



Attribute name	Attribute value
com.ibm.wmqfte.DestinationAgent	MUMBAI.AGENT
com.ibm.wmqfte.MqmdUser	MUSR_MQADMIN
com.ibm.wmqfte.OriginatingHost	192.168.65.131
com.ibm.wmqfte.OriginatingUser	wmbadmin
com.ibm.wmqfte.SourceAgent	WASH.AGENT
com.ibm.wmqfte.TransferId	414d512057415348514d202020202020f2f7ed4920005...

Figure 10-20 Metadata view in one transfer log properties

3. Switch to the XML view, as shown in Figure 10-21 on page 285. Additional information is displayed in XML format.



```
# STARTED

<?xml version="1.0" encoding="UTF-8"?>
<transaction ID="414d512057415348514d202020202020f2f7ed4920005c02" agentRole="sourceAgent" version="1.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="TransferLog.xsd">
<action time="2009-04-21T18:58:21.624Z">started</action>
<sourceAgent QMgr="WASHQM" agent="WASH.AGENT">
<systemInfo architecture="x86" name="Windows XP" version="5.1 build 2600 Service Pack 3"/>
</sourceAgent>
<destinationAgent QMgr="MUMBAIQM" agent="MUMBAI.AGENT"/>
<originator>
<hostName>192.168.65.131</hostName>
<userID>wmbadmin</userID>
<mqmdUserID>MUSR_MQADMIN</mqmdUserID>
</originator>
<transferSet startTime="2009-04-21T18:58:21.624Z" total="1">
<metaDataSet>
<metaData key="com.ibm.wmqfte.SourceAgent">WASH.AGENT</metaData>
<metaData key="com.ibm.wmqfte.DestinationAgent">MUMBAI.AGENT</metaData>
<metaData key="com.ibm.wmqfte.OriginatingHost">192.168.65.131</metaData>
<metaData key="com.ibm.wmqfte.MqmdUser">MUSR_MQADMIN</metaData>
<metaData key="com.ibm.wmqfte.TransferId">414d512057415348514d202020202020f2f7ed4920005c02</metaData>
<metaData key="com.ibm.wmqfte.OriginatingUser">wmbadmin</metaData>
</metaDataSet>
</transferSet>
</transaction>

----

# IN PROGRESS

<?xml version="1.0" encoding="UTF-8"?>
<transaction ID="414d512057415348514d202020202020f2f7ed4920005c02" agentRole="sourceAgent" version="1.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="TransferLog.xsd">
<action time="2009-04-21T18:58:22.406Z">progress</action>
<sourceAgent QMgr="WASHQM" agent="WASH.AGENT">
<systemInfo architecture="x86" name="Windows XP" version="5.1 build 2600 Service Pack 3"/>
</sourceAgent>
<destinationAgent QMgr="MUMBAIQM" agent="MUMBAI.AGENT">
<systemInfo architecture="x86" name="Windows XP" version="5.1 build 2600 Service Pack 3"/>
</destinationAgent>
<originator>
<hostName>192.168.65.131</hostName>
<userID>wmbadmin</userID>
<mqmdUserID>MUSR_MQADMIN</mqmdUserID>
</originator>
<transferSet index="0" size="1" startTime="2009-04-21T18:58:21.624Z" total="1">
<item mode="binary">
<source disposition="leave">
<file>C:\Creature.jpg</file>
<checksum method="none"/>
</source>
<destination exist="error">
<file>C:\files\Creature.jpg</file>
<checksum method="none"/>
</destination>
<status resultCode="0"/>
</item>
</transferSet>
</transaction>
```

Figure 10-21 The transfer history details in XML format

10.3.4 Subscriptions

WebSphere MQ File Transfer Edition provides a flexible way to monitor and review the file transfer state and history in an FTE network. You can receive publications from Agents about all file transfers by *subscribing* your own queue in the Subscriptions view in WebSphere MQ Explorer. When a coordination queue manager publishes transfer log messages, your queue receives the real-time XML messages, and these messages can be processed using your own program. The following steps provide an example of the process.

To subscribe a queue:

1. Open WebSphere MQ Explorer, and right-click **IBM WebSphere MQ** → **Queue Managers** → **qMgrName** → **Subscriptions** → **New** → **Subscription**, as shown in Figure 10-22.

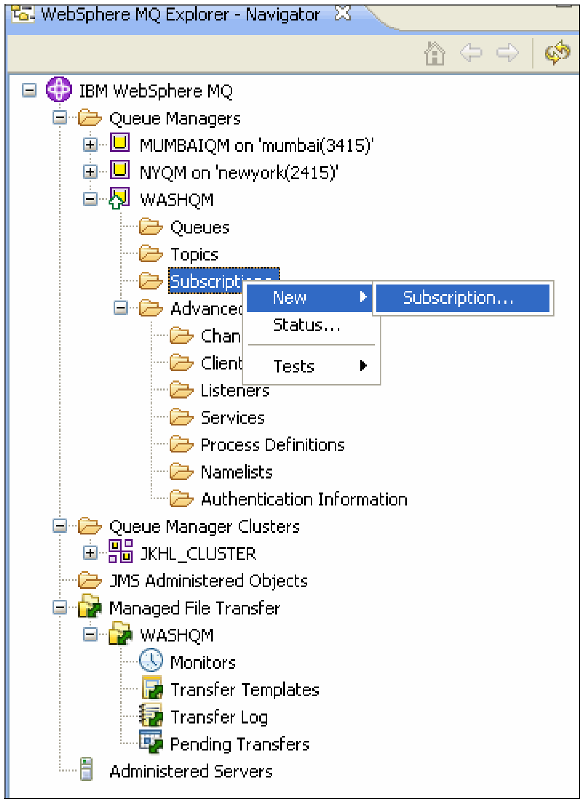
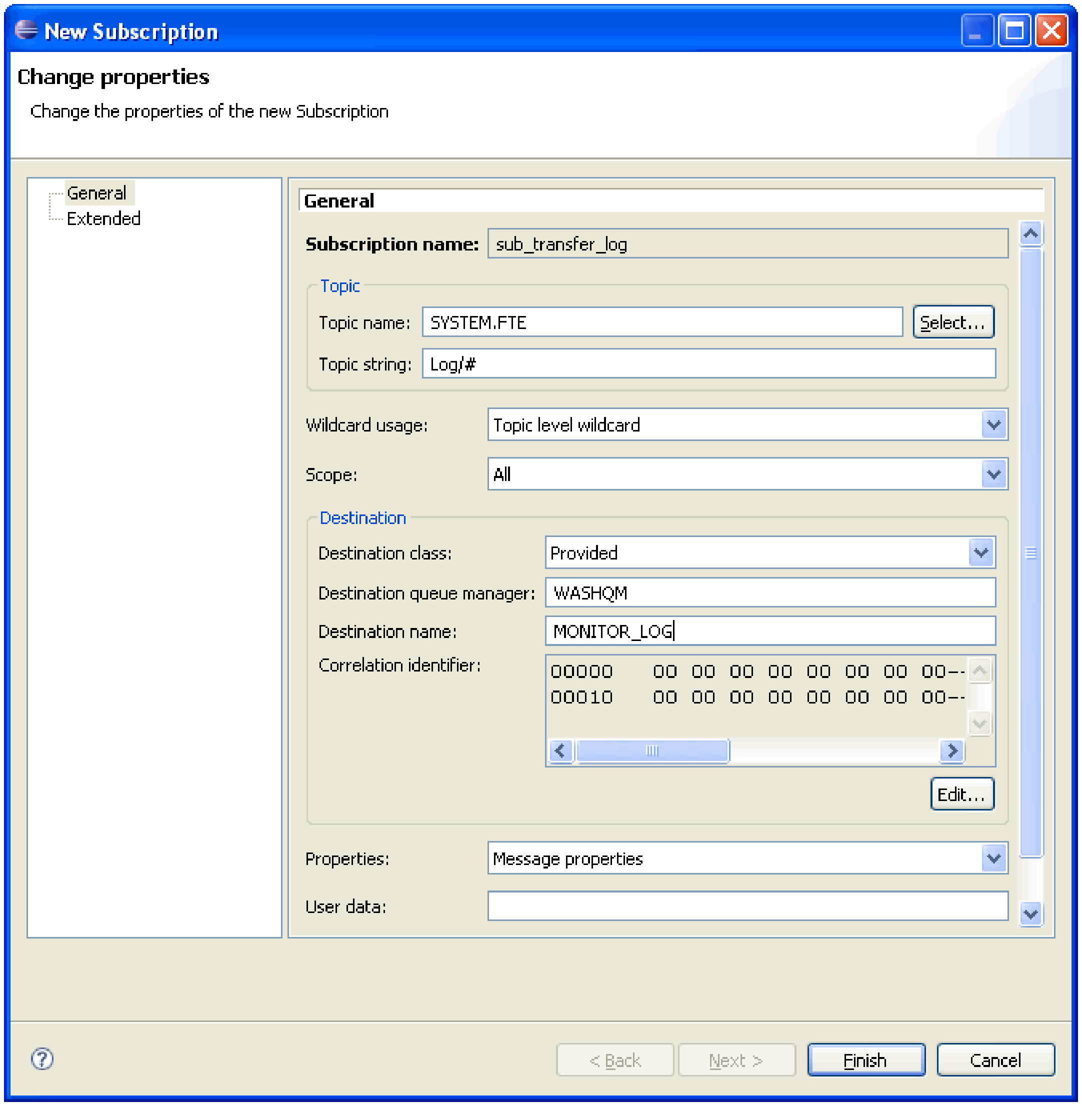


Figure 10-22 Creating a new subscription

qMgrName: qMgrName is the name of the coordination queue manager from which your queue subscriber receives transfer log messages.

2. Input subscription information, such as topic name, topic string, destination queue manager, and destination name, as seen in Figure 10-23. Click **Finish**.


New Subscription

Change properties

Change the properties of the new Subscription

General

Subscription name: sub_transfer_log

Topic

Topic name: SYSTEM.FTE

Topic string: Log/#

Wildcard usage: Topic level wildcard

Scope: All

Destination

Destination class: Provided

Destination queue manager: WASHQM

Destination name: MONITOR_LOG

Correlation identifier:

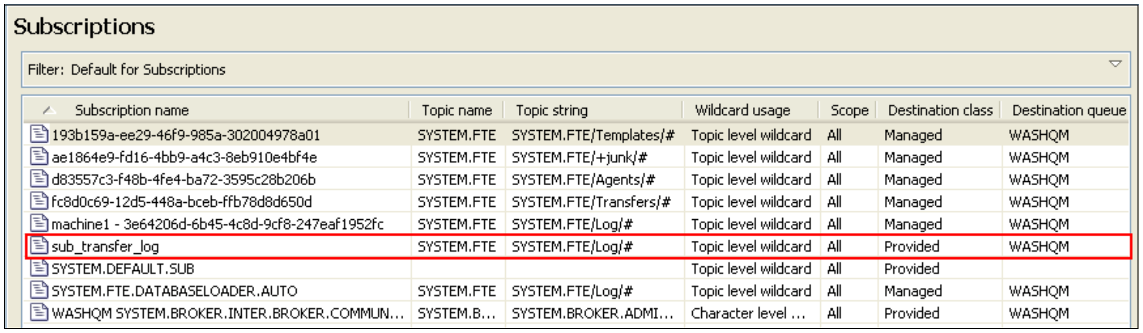
```
00000     00  00  00  00  00  00  00  00--
00010     00  00  00  00  00  00  00  00--
```

Properties: Message properties

User data:

Figure 10-23 Input subscription information

3. Check your subscription in the WebSphere MQ Explorer, as shown in Figure 10-24.


Subscriptions

Filter: Default for Subscriptions

Subscription name	Topic name	Topic string	Wildcard usage	Scope	Destination class	Destination queue
193b159a-ee29-46f9-985a-302004978a01	SYSTEM.FTE	SYSTEM.FTE/Templates/#	Topic level wildcard	All	Managed	WASHQM
ae1864e9-fd16-4bb9-a4c3-8eb910e4bf4e	SYSTEM.FTE	SYSTEM.FTE/+junk/#	Topic level wildcard	All	Managed	WASHQM
d83557c3-f48b-4fe4-ba72-3595c28b206b	SYSTEM.FTE	SYSTEM.FTE/Agents/#	Topic level wildcard	All	Managed	WASHQM
fc8d0c69-12d5-448a-bceb-ffb78d8d650d	SYSTEM.FTE	SYSTEM.FTE/Transfers/#	Topic level wildcard	All	Managed	WASHQM
machine1 - 3e64206d-6b45-4c8d-9cf8-247eaf1952fc	SYSTEM.FTE	SYSTEM.FTE/Log/#	Topic level wildcard	All	Managed	WASHQM
sub_transfer_log	SYSTEM.FTE	SYSTEM.FTE/Log/#	Topic level wildcard	All	Provided	WASHQM
SYSTEM.DEFAULT.SUB			Topic level wildcard	All	Provided	
SYSTEM.FTE.DATABASELOADER.AUTO	SYSTEM.FTE	SYSTEM.FTE/Log/#	Topic level wildcard	All	Managed	WASHQM
WASHQM SYSTEM.BROKER.INTER.BROKER.COMMUN...	SYSTEM.B...	SYSTEM.BROKER.ADMI...	Character level ...	All	Provided	WASHQM

Figure 10-24 Your subscription from SYSTEM.FTE topic

You can write WebSphere MQ subscribing applications to receive the XML format transfer logs without queues.

10.4 Monitoring scheduled file transfers with pending state

You can view scheduled file transfers using the Pending Transfers view in WebSphere MQ Explorer, as shown in Figure 10-25.

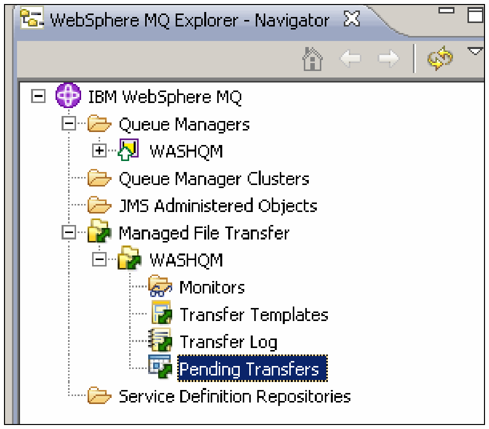
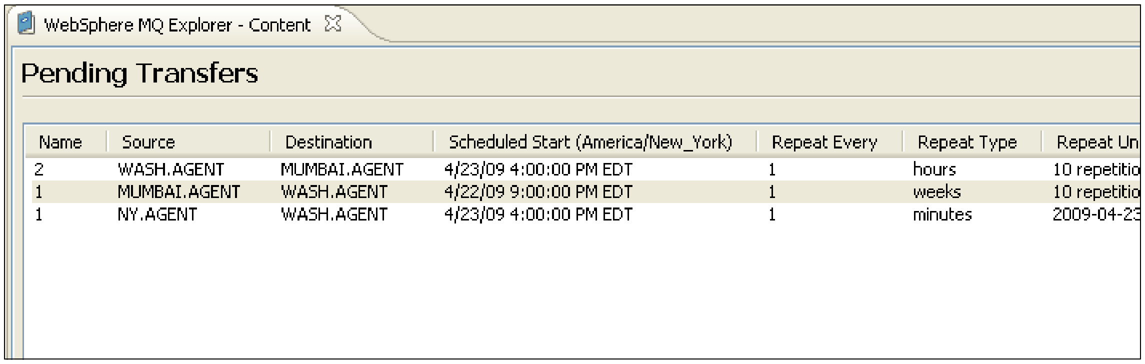


Figure 10-25 Pending Transfers view

All the transfers that are scheduled but not yet started, or are started and in progress, are shown in the Pending Transfers view, as shown in Figure 10-26 on page 289.



Pending Transfers

Name	Source	Destination	Scheduled Start (America/New_York)	Repeat Every	Repeat Type	Repeat Un
2	WASH.AGENT	MUMBAI.AGENT	4/23/09 4:00:00 PM EDT	1	hours	10 repetitio
1	MUMBAI.AGENT	WASH.AGENT	4/22/09 9:00:00 PM EDT	1	weeks	10 repetitio
1	NY.AGENT	WASH.AGENT	4/23/09 4:00:00 PM EDT	1	minutes	2009-04-23

Figure 10-26 Pending transfers list

In this view, to cancel a scheduled file transfer in pending state, right-click the transfer, and select **Cancel**.

10.5 Archiving audit information in the database

WebSphere MQ File Transfer Edition provides an additional software package called the DatabaseLogger to store audit information in an SQL database.

When FTE transfers files, it publishes information about its actions to a topic on the coordination queue manager. You can use the Database Logger to copy this information into a SQL table for analysis and auditing purposes.

The Database Logger is a standalone Java application that is installed on a machine that hosts the coordination queue manager and the database. It connects to the local coordination queue manager and to a DB2 or Oracle database using the Type 2 JDBC™ driver. This type of connection is necessary because the Database Logger uses the queue manager's XA-transaction support to coordinate a global transaction over both queue manager and database, which protects the data. On z/OS, the Database Logger uses the Resource Recovery Service (RRS) to coordinate transactions, and on that platform only DB2 is supported.

10.5.1 System requirements and installation

The operating system and database requirements are:

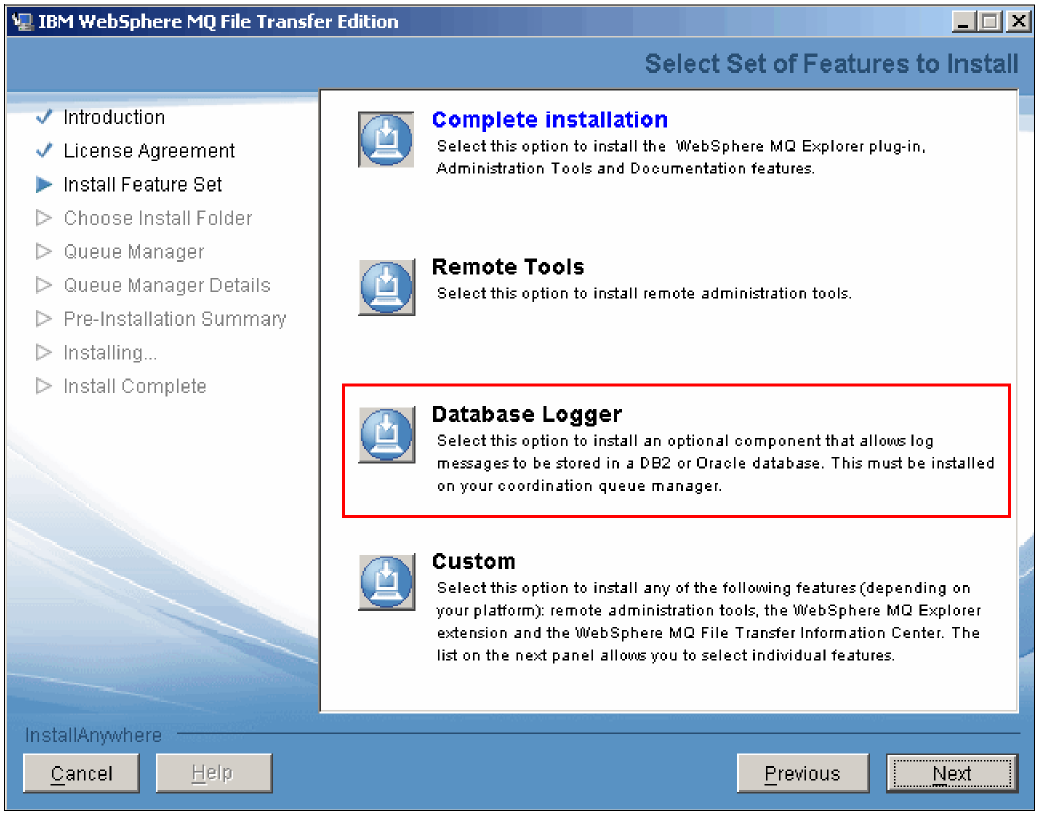
Database Logger operating system requirements:

- AIX®, HPUX, Solaris™, Windows

Database Logger SQL DB requirements:

- IBM DB2 V9.1 for UNIX and Windows
- IBM DB2 V9.5 for UNIX and Windows
- Oracle 10g Release 2

The Database Logger is an optional component of FTE, and if you want to use it, you can get it from the Remote Tools CD. During the installation, choose this component in the `Select Set of Features to install` window or click **Complete installation**, as seen in Figure 10-27.



Complete installation
Select this option to install the WebSphere MQ Explorer plug-in, Administration Tools and Documentation features.

Remote Tools
Select this option to install remote administration tools.

Database Logger
Select this option to install an optional component that allows log messages to be stored in a DB2 or Oracle database. This must be installed on your coordination queue manager.

Custom
Select this option to install any of the following features (depending on your platform): remote administration tools, the WebSphere MQ Explorer extension and the WebSphere MQ File Transfer Information Center. The list on the next panel allows you to select individual features.

Figure 10-27 Database logger installation window

10.5.2 Configuring the Database Logger

Before you can start the Database Logger command, you must:

1. Create database and tables with SQL scripts.
2. Set user permissions.
3. Configure transaction support in queue manager.
4. Edit the databaselogger.properties file.

For more configuration information, see the *Using the database logger* section in the WebSphere MQ File Transfer Edition Information Center, which is located at:

`http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp`

10.5.3 Starting the Database Logger

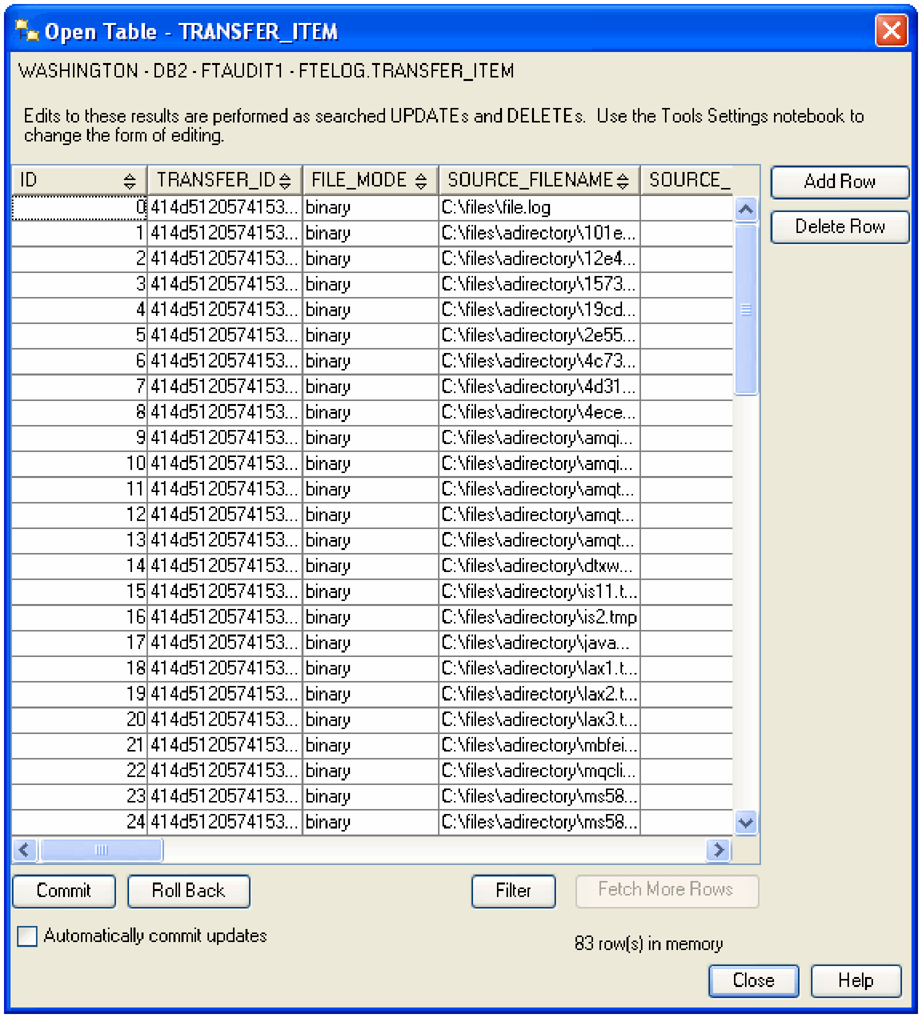
After all of the configurations are ready, you can start the Database Logger using the **`fteStartDatabaseLogger`** command. By default, the Database Logger runs in the background, and it places its output in a file in the logs directory. If you want to run the Database Logger in the foreground, and have its output sent to the console and to the log file, add the `-F` switch to the **`fteStartDatabaseLogger`** command.

Note: The **`fteStartDatabaseLogger`** command is not in the bin directory of the FTE basic installation because it is an optional component. You must change to the bin directory of the Remote Tools installation at <WMQFTE_install_directory>/tools/bin.

- When the file transfers are running, the auditing and logging information is received and stored into a database by the Database Logger. In the FTE log database, there are tables for basic transfers, scheduled transfers, file trigger condition, metadata, and transfer calling action. The following tables are provided:
  - TRANSFER_EVENT
  - TRANSFER
  - TRANSFER_ITEM
  - SCHEDULE_ACTION
  - SCHEDULE_SPEC
  - SCHEDULE_ITEM
  - SCHEDULE
  - TRIGGER_CONDITION
  - METADATA
  - TRANSFER_CALLS
  - CALL_RESULT

– CALL_ARGUMENT
– CALL_REQUEST
– CALL

The Database Logger helps to *harden* the transfer history information in the DB2 or Oracle database. However it does not provide a Web or any other interface to display these records when users need to search the history transfer information. If you need to search the history transfer information, you can use the database command interface and graphic management tool or you must customize the Web interfaces for easy auditing. The auditing and logging data is shown in the DB2 Control Center in Figure 10-28.



WASHINGTON - DB2 - FTAUDIT1 - FTELOG.TRANSFER_ITEM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

ID	TRANSFER_ID	FILE_MODE	SOURCE_FILENAME	SOURCE_
0	414d5120574153...	binary	C:\files\file.log	
1	414d5120574153...	binary	C:\files\adirectory\101e...	
2	414d5120574153...	binary	C:\files\adirectory\12e4...	
3	414d5120574153...	binary	C:\files\adirectory\1573...	
4	414d5120574153...	binary	C:\files\adirectory\19cd...	
5	414d5120574153...	binary	C:\files\adirectory\2e55...	
6	414d5120574153...	binary	C:\files\adirectory\4c73...	
7	414d5120574153...	binary	C:\files\adirectory\4d31...	
8	414d5120574153...	binary	C:\files\adirectory\4ece...	
9	414d5120574153...	binary	C:\files\adirectory\amqi...	
10	414d5120574153...	binary	C:\files\adirectory\amqi...	
11	414d5120574153...	binary	C:\files\adirectory\amqt...	
12	414d5120574153...	binary	C:\files\adirectory\amqt...	
13	414d5120574153...	binary	C:\files\adirectory\amqt...	
14	414d5120574153...	binary	C:\files\adirectory\dtxw...	
15	414d5120574153...	binary	C:\files\adirectory\is11.t...	
16	414d5120574153...	binary	C:\files\adirectory\is2.tmp	
17	414d5120574153...	binary	C:\files\adirectory\java...	
18	414d5120574153...	binary	C:\files\adirectory\lax1.t...	
19	414d5120574153...	binary	C:\files\adirectory\lax2.t...	
20	414d5120574153...	binary	C:\files\adirectory\lax3.t...	
21	414d5120574153...	binary	C:\files\adirectory\mbfei...	
22	414d5120574153...	binary	C:\files\adirectory\mqcli...	
23	414d5120574153...	binary	C:\files\adirectory\ms58...	
24	414d5120574153...	binary	C:\files\adirectory\ms58...	

☐ Automatically commit updates

83 row(s) in memory

Figure 10-28 File transfer list shown in the TRANSFER_ITEM table

10.6 Summary

In this chapter, we discussed:

- How to monitor FTE file transfers that are in progress.
- How to view file transfer state and history information in WebSphere MQ Explorer.
- Using the WebSphere MQ Explorer, we described how to customize the transfer log and audit information.
- The Database Logger optional component.

11

Recovering and troubleshooting

In this chapter, we discuss WebSphere MQ File Transfer Edition recovery, restarting transfers, and troubleshooting WebSphere MQ File Transfer Edition.

The topics that we discuss in this chapter are:

11.1 Recovering from system failures

WebSphere MQ File Transfer Edition maintains all necessary restart and recovery information as persistent messages in its queues. It does not store any of that information in memory. WebSphere MQ File Transfer Edition relies on WebSphere MQ to maintain the status messages for recovery from failures. During restart it accesses its queues obtaining the status information of transfers that were in progress. To know the current condition of the files that are being transferred it gathers information from the operating systems file service.

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

- Typically, if there is a problem while a file is in the process of being transferred, WebSphere MQ File Transfer Edition 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 the agent is restarted. If an agent looses the connection to its agent queue manager, the agent waits in an infinite loop trying to reconnect to the queue manager. When the agent successfully reconnects to its queue manager, the current transfer continues.

You can check the status of your transfers in the WebSphere MQ File Transfer Edition WebSphere MQ Explorer plug-in. 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 that are involved in the transfer. The stalled status appears in the Current Transfer Progress window in the Rate column.

11.2 Troubleshooting WebSphere File Transfer Edition

In this section, we provide troubleshooting information for:

- WebSphere MQ File Transfer Edition issues
- WebSphere MQ File Transfer Edition agent trace level
- WebSphere MQ diagnostics

11.2.1 WebSphere MQ File Transfer Edition issues

Use the following reference information to help you to diagnose errors in WebSphere MQ File Transfer Edition:

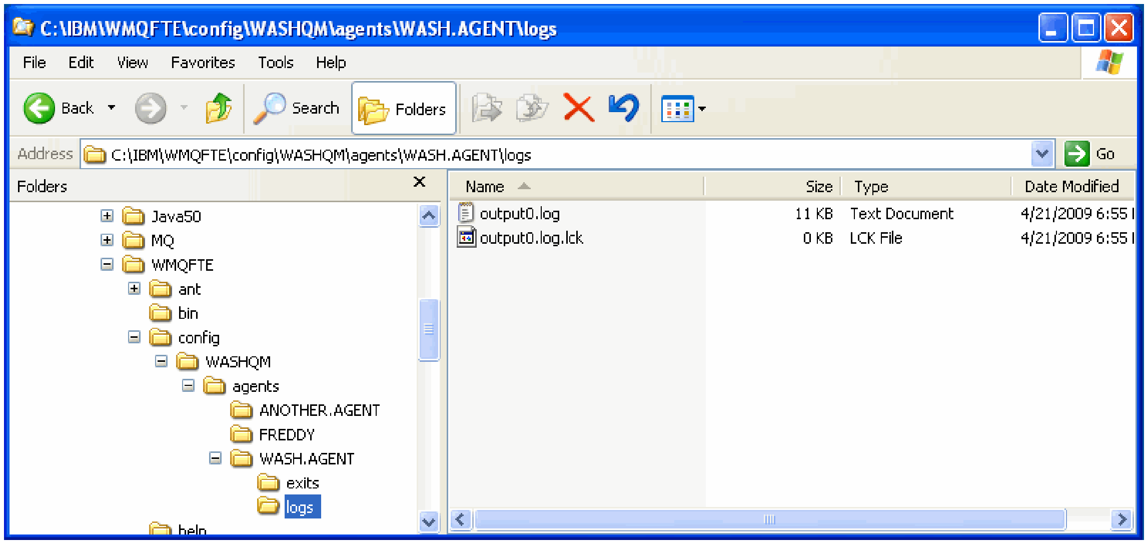
- WebSphere MQ File Transfer Edition agent event logs
- WebSphere MQ File Transfer Edition bindings to WebSphere MQ v6
- WebSphere MQ File Transfer Edition exit multiple executions
- Client re-connection issues

WebSphere MQ File Transfer Edition diagnostic message is described in the WebSphere MQ File Transfer Edition Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.messages.doc/messages_main.htm

WebSphere MQ File Transfer Edition agent event logs

WebSphere MQ File Transfer Edition maintains logs that you can review to see what occurred during the execution of commands. The agent event log records events that the agent generates and can be useful when diagnosing transfer problems. However it does not show directly what happened during the execution of a command. When executing an FTE command, the command output by default is sent to standard output. Figure 11-1 shows the directory location.



Name	Size	Type	Date Modified
output0.log	11 KB	Text Document	4/21/2009 6:55
output0.log.lck	0 KB	LCK File	4/21/2009 6:55

Figure 11-1 WebSphere MQ File Transfer Edition log directory

WebSphere MQ File Transfer Edition bindings to WebSphere MQ V6

If you are using bindings mode to connect to a WebSphere MQ Version 6.0 queue manager for either an agent or for commands and are having problems, ensure that you applied fix pack 6.0.2.5 to WebSphere MQ Version 6.0.

WebSphere MQ File Transfer Edition exit multiple executions

If you are using user exit routines and there is a failure while the user exit is being called or just after the exit is called, for example a product failure or power cut, it is possible that the user exit will be called more than once.

Client re-connection issue

If you have an agent with a queue manager on a machine with an IP address that is assigned using DHCP (rather than a static IP address), and the agent connects to that machine using a client TCP/IP connection, you must start the agent with the following system environment variable set:

- On Windows:

```
set FTE_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=<value>"
```

- On UNIX:

```
export FTE_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=<value>"
```

In the system environment variable sets, `<value>` is the time interval in seconds between each flush of the JVM's cached DNS values. If the IP address of the queue manager machine is reassigned for any reason (for example, because of a network outage, an IP lease expiry, or a machine 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 because the JVM internally caches the IP addresses of host names and does not refresh them by default.

11.2.2 WebSphere MQ File Transfer Edition agent trace level

Use the **`fteSetAgentTraceLevel`** command to dynamically modify the current trace level for an agent. This command switches agent trace on and off or changes 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.

For information about how to set the agent trace level, see 5.4.11, “Setting agent trace level” on page 139.

WebSphere MQ File Transfer Edition logs

The trace files that are produced are located in:

<configuration_directory>\<coordination_qmgr_name>\agents\<agent_name>\logs

Figure 11-2 shows this location on a Linux system.

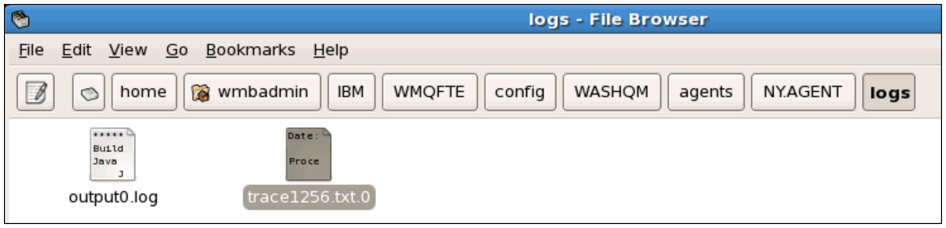


Figure 11-2 Trace Log location Linux system

Because running trace can impact your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary.

Table 11-1 shows WebSphere MQ File Transfer Edition trace values. These are agent properties.

Table 11-1 WebSphere MQ File Transfer Edition trace values

Name	Properties function	Value
traceFiles	The total number of trace files to keep.	5
traceSize	The maximum size in MB for each trace file, before trace wraps onto the next file.	20
trace	Trace specification when agent is to be run with trace enabled at start up.	None
ITLMTraceLevel	ITLM trace level. Controls the amount of trace and log information that is output by Tivoli® License Manager. Possible values (in increasing level of verboseness) are: MIN, MID, and MAX	MIN

The Trace logs are text files and can be read by text editors.

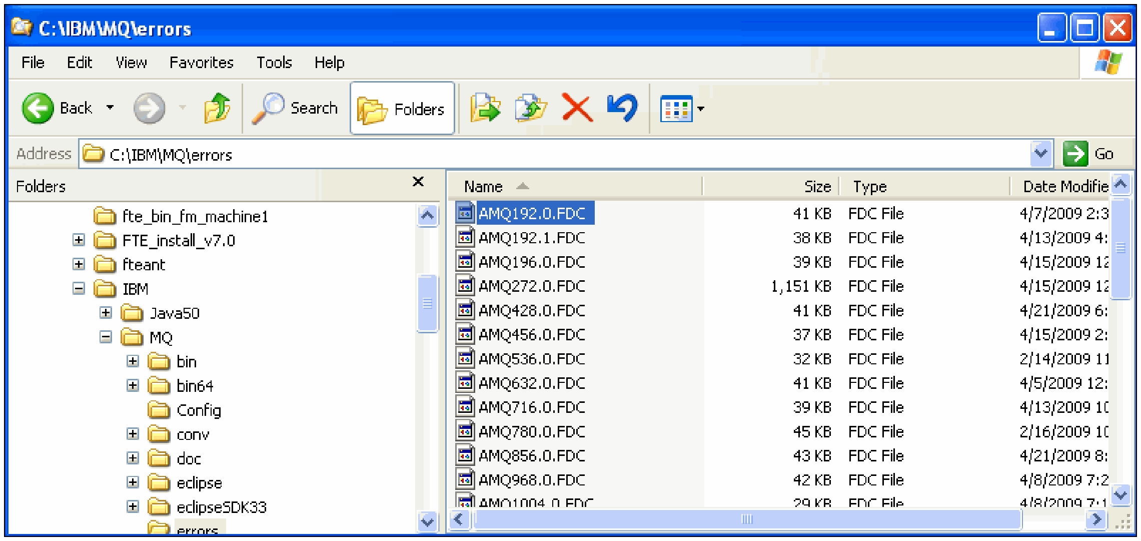
11.2.3 WebSphere MQ diagnostics

The WebSphere MQ logs can be a useful starting point for investigating problems.

On the z/OS system there are messages written to each of the address spaces syslog for the queue manager or the system's syslog. In addition to the syslogs, there are other z/OS diagnostic tools that your systems programmer can help you work with.

On open systems, WebSphere MQ has error directories where information is written that might help resolve problems.

WebSphere MQ has product-level errors recorded, as shown in Figure 11-3.



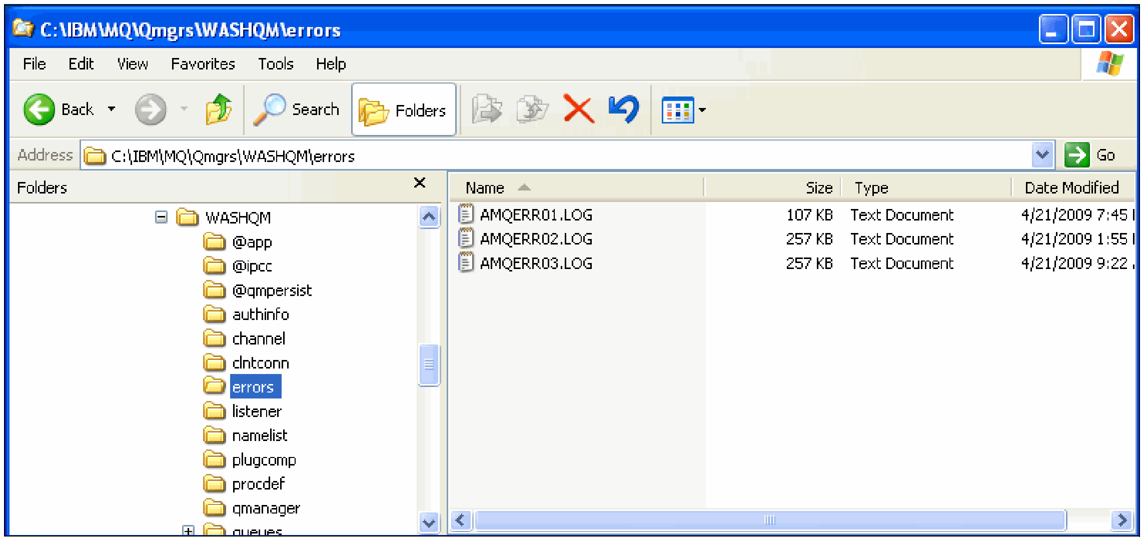
Address: C:\IBM\MQ\errors

- fte_bin_fm_machine1
- FTE_install_v7.0
- fteant
- IBM
  - Java50
  - MQ
    - bin
    - bin64
    - Config
    - conv
    - doc
    - eclipse
    - eclipseSDK33
    - errors

Name	Size	Type	Date Modifie
AMQ192.0.FDC	41 KB	FDC File	4/7/2009 2:3
AMQ192.1.FDC	38 KB	FDC File	4/13/2009 4:
AMQ196.0.FDC	39 KB	FDC File	4/15/2009 12
AMQ272.0.FDC	1,151 KB	FDC File	4/15/2009 12
AMQ428.0.FDC	41 KB	FDC File	4/21/2009 6:
AMQ456.0.FDC	37 KB	FDC File	4/15/2009 2:
AMQ536.0.FDC	32 KB	FDC File	2/14/2009 11
AMQ632.0.FDC	41 KB	FDC File	4/5/2009 12:
AMQ716.0.FDC	39 KB	FDC File	4/13/2009 10
AMQ780.0.FDC	45 KB	FDC File	2/16/2009 10
AMQ856.0.FDC	43 KB	FDC File	4/21/2009 8:
AMQ968.0.FDC	42 KB	FDC File	4/8/2009 7:2
AMQ1004.0.FDC	29 KB	FDC File	4/8/2009 7:1

Figure 11-3 WebSphere MQ level errors

WebSphere MQ queue manager errors are located in their own directory, as shown in Figure 11-4 on page 301.



```
Name              Size    Type            Date Modified
AMQERR01.LOG    107 KB    Text Document   4/21/2009 7:45
AMQERR02.LOG    257 KB    Text Document   4/21/2009 1:55
AMQERR03.LOG    257 KB    Text Document   4/21/2009 9:22
```

Figure 11-4 WebSphere MQ queue manager level error directory

12

Security considerations

In this chapter, we discuss security concepts that apply to WebSphere MQ File Transfer Edition. Each organization has its own security requirements. Use the discussions in this chapter as guidance when working with your WebSphere MQ and operating system administrators.

The topics that we discuss in this chapter are:

This chapter does not address the SSL support that is provided with WebSphere MQ File Transfer Edition. For information about using the SSL support with WebSphere MQ File Transfer Edition, see Chapter 7, “Phase 1: Basic file transfers” on page 159.

12.1 Introduction to WebSphere MQ File Transfer Edition security

Directly after installation and with no modification, WebSphere MQ File Transfer Edition is unsecured. In a production environment, you must give consideration to:

- Controlling who can invoke file transfer operations
- Access to files granted to WebSphere MQ File Transfer Edition agents.
- Who can read and write the files before and after transfers
- How to protect the integrity of files

WebSphere MQ File Transfer Edition agents work with two resource managers:

- WebSphere MQ resource manager

  Provides messaging services for the WebSphere MQ File Transfer Edition agents.

- Operating system resource manager

  Provides authentication of users and their file and directory service access.

WebSphere MQ File Transfer Edition administrators working with WebSphere MQ administrators and the operating system's security administrators must implement security that bridges both domains.

12.2 WebSphere MQ File Transfer Edition, WebSphere MQ security

WebSphere MQ File Transfer Edition agents are like any other application when accessing WebSphere MQ: they must be identified, authenticated, and assigned a userID.

WebSphere MQ is a messaging, publishing and subscription engine that has many intercommunication options. You must analyze and secure each of these channels so that their security meets your needs.

12.2.1 Creating WebSphere MQ File Transfer Edition user IDs and groups

The authorization for users, administrators, and applications is based on userIDs and membership of groups. Either the userID or the groups to which a userID is associated need the permission to perform the actions that are being requested on an object.

When starting the WebSphere MQ File Transfer Edition agents, do not start them with a userID that has WebSphere MQ administrator or *super user* authority. The danger with those authorities is allowing a transfer to access and modify critical files. The groups that the agents belong to also should not provide those authorities.

Instead of granting authority to the individual userIDs for all of the various objects that might be involved, consider configuring security groups for the purposes of administering WebSphere MQ File Transfer Edition access control. Group definitions might be as simple as two groups FTEUSER and FTEAGENT or as complex as groups for each business application PAYROLL, FINANCE, or INVENTORY.

12.2.2 Access to WebSphere MQ file objects

WebSphere MQ determines the access authority of WebSphere MQ objects. On z/OS systems it uses the Security Access Facility (SAF) to determine access rights. On distributed systems, WebSphere MQ uses different methods to determine access rights depending on the operating system. WebSphere MQ object access checks are made for the requesting application's authorities, typically based on the requester userID but can be done using alternate userID authority.

WebSphere MQ connections

The WebSphere MQ connection is the first level of security to the WebSphere MQ File Transfer Edition network domain. It is critical that these connections are secure.

Commands that are run by operational users, administrative users, and the WebSphere MQ File Transfer Edition plug-in that is in WebSphere MQ Explorer must connect to a command queue manager. The agent processes must connect to the agent queue manager. If you created the FTE groups, then grant the FTEUSER group connect authority for the command queue manager and the FTEAGENT group connect authority to the agent queue manager.

The WebSphere MQ File Transfer Edition agents connect to their queue managers using two methods, bindings or client. UserID determination and what userID is contained in messages depends on how the agent is connected to its queue manager. Messages from bindings-connected agents contain the userID that the agent is running under and is determined by the operating system.

Messages from client-connected agents contain the MCAUSER value, which is determined by the WebSphere MQ SVRCONN.

You should provide an initial MCAUSER value with low or no authority on the channel that the agent uses to connect to its queue manager. You can modify the userID using a channel exit of the receiving channel to limit the access rights. It might be a good idea to have every agent have a unique ID within the WebSphere MQ File Transfer Edition network domain.

For both client and bindings connected agents, you can use the property publicationMDUser (in the agent.properties file) to specify a userID, 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 userID, and you give this permission by granting setid authority to the userID that the agent runs under.

Table 12-1 summarizes the access control configuration for the FTEUSER and FTEAGENT groups in the security connection scheme.

Table 12-1 Access control configuration in the security connection scheme

Object	Object type	FTEUSER	FTEAGENT
Agent queue manager	Queue manager		CONNECT and SETID
Coordination queue manager	Queue manager	CONNECT	
Queue manager	Queue manager	CONNECT	CONNECT

WebSphere MQ queues

The WebSphere MQ File Transfer Edition queues must be protected from unauthorized manipulations, such as MQGETing, MQPUTing, deletion, or alteration. This protection includes access by the WebSphere MQ channels.

The agent's command queue must be available to any user who is authorized to create new file transfer requests or issue commands. To satisfy this requirement, only grant the FTEUSER group *put access* to the SYSTEM.FTE.COMMAND queue. Grant the FTEAGENT group *put and get access* to the agent's SYSTEM.FTE.COMMAND queue. Only WebSphere MQ File Transfer Edition agents need to use the DATA, STATE, EVENT, and REPLY WebSphere MQ File Transfer Edition system queues; therefore, only grant the FTEAGENT group *put and get access*.

Administrative commands, for example the **`fteStartAgent`** command and the WebSphere MQ Explorer, must put messages to the agent's SYSTEM.FTE.COMMAND queue. Because commands might come across message channels, the queue managers on the sending side should be trusted, or you might need to use a PUTAUT other than default on your receiving channels.

Synchronous file transfer requests wait for the transfer to complete and rely on a temporary reply queue being created and populated. Therefore grant any user that can run a synchronous file transfer command, *display, put, get, and browse authorities* on the dynamic queues and *browse* on model queue definition as it is known to the agent. By default this is SYSTEM.DEFAULT.MODEL.QUEUE, but you can change the name by setting values for the properties 'modelQueueName' and 'dynamicQueuePrefix' in the command.properties file.

12.2.3 WebSphere MQ publish and subscribe

Only the agent process needs to place messages on the SYSTEM.FTE queue for publication to the SYSTEM.FTE topic. Grant *publish* and *subscribe authority* to the FTEAGENT group. 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 that is contained in the message descriptor structure (MQMD) of the message. Remember that those messages might come across message channels and those channels properties must be reviewed.

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.

Access to the coordination function of WebSphere MQ File Transfer Edition is only supported on one queue manager in the network domain. All users who want published information must be connected to that queue manager to subscribe to it. These restrictions therefore reduce security concerns.

Table 12-2 summarizes the access control configuration for FTEUSER and FTEAGENT in the security queue access scheme.

Table 12-2 Access control configuration in the security queue access scheme

Object	Object type	FTEUSER	FTEAGENT
SYSTEM.FTE	Local queue		PUT and GET
SYSTEM.FTE.COMMAND	Local queue	PUT	PUT and GET
SYSTEM.FTE.DATA	Local queue		PUT and GET
SYSTEM.FTE.EVENT	Local queue		PUT and GET
SYSTEM.FTE.REPLY	Local queue		PUT and GET

Object	Object type	FTEUSER	FTEAGENT
SYSTEM.FTE.STATE	Local queue		PUT and GET
SYSTEM.FTE	Local topic		PUBLISH and SUBSCRIBE
SYSTEM.DEFAULT.MODEL.QUEUE (or the model queue defined in WebSphere MQ File Transfer Edition that is used to create a temporary reply queue)	Model queue	DISPLAY, PUT, GET, and BROWSE	DISPLAY, PUT, GET, and BROWSE

12.2.4 WebSphere MQ channels

WebSphere MQ uses channels to move messages to the destination queues. In the case of client connection, it also moves information to the client application.

The challenge is in securing remotely connected users and connections from other queue managers. Any inbound channel that does not have a low-privileged account in the MCAUSER attribute might be capable of administering the queue manager and the FTE agent.

The first step in securing any queue manager is to disable any inbound channels that should not legitimately be used, by setting MCAUSER('nobody') for any SYSTEM.DEF.*, SYSTEM.AUTO.*, SYSTEM.ADMIN.SVRCONN channels.

Client channels

Clients that connect to a queue manager use a server connection channel, better know as a SVRCONN. There is a SYSTEM.DEF.SVRCONN that could be used; however, some organizations do not allow connections to it. An organization might create application-specific Sovereigns, such as FTEAGENT.SRVCONN or FTEUSER.SVRCONN. In either case, the client connection can use SSL to secure access to a queue manager from a client. WebSphere MQ File Transfer Edition does not use a client channel table to provide information about the connections; instead, it obtains that information from the .properties files for each agent's functions.

The strategy for securing client connections is to strongly authenticate the connection request, and then ensure that an MCAUSER value corresponds to the authenticated identity.

These connections must be authenticated using SSL certificates, and a channel security exit is then used to dynamically map the SSL certificate to an

MCAUSER value for that channel instance. In this way, many users can simultaneously and securely access a single channel definition using a variety of MCAUSER values. In this configuration, the channel is normally defined with MCAUSER ('nobody') or another low-privileged account, so that if the exit fails, the channel is not exposing administrative privileges. The same level of security can be obtained without using an exit by judicious use of SSLPEER and SSLCAUTH on the channel and a static MCAUSER value. With this strategy, you must define multiple SVRCONN channels, one for each security role.

Message and cluster channels

You must also secure inbound MCA channels, including channels of type RCVR, RQSTR, and CLUSRCVR, because these channels expose administrative access if left in their default settings. These channels could also use channel SSL and MCAUSER value to restrict the channels' access to queues.

In a cluster, this configuration requires both a Channel Auto-Definition exit and a Security exit. The CHAD exit configures the security exit which, in turn, sets the MCAUSER attribute of the channel to the appropriate service account.

12.3 WebSphere MQ File Transfer Edition operating systems security

WebSphere MQ File Transfer Edition operates on multiple platforms, and each of these operating systems has its own style of security mechanisms. WebSphere MQ File Transfer Edition works with the file system of the operating system.

WebSphere MQ File Transfer Edition requires access to the files that it is transferring, which means that without any security, every file on every agent system potentially makes up the WebSphere MQ File Transfer Edition file domain. The operating system file security should restrict access by the agents to files they are not transferring or should not transfer. In addition to file security, WebSphere MQ File Transfer Edition implements a *sandbox* file access concept.

The authority that the agent process runs under determines the files that the agent can read and write from the file system. How the authority is configured is system dependent. If you have an FTEAGENT group, add the userID that the agent process runs under to that group.

12.3.1 File and directory security

For any file transfer request, the agent processes require some level of access to their local file systems. To transfer a file, the userID that the source agent runs under must have read access to the source file. Additionally, you might need to give the source agent delete authority depending on the source disposition attribute. The destination agent process 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.

However, we do not want to grant access to any more files than are needed by WebSphere MQ File Transfer Edition agents to perform the transfers.

The agents must be the only ones that can access the WebSphere MQ File Transfer Edition executable and the configuration files. An exception being the WebSphere MQ File Transfer Edition administrators. WebSphere MQ File Transfer Edition allows commands and scripts to be executed before and after a file is transferred. Those executables must be protected from unauthorized access.

The optimum relation to the files is that both the agent and the applications can read, write, and delete their files.

With z/OS, access authority is based on qualifiers in the file name, and it is relatively easy to create profiles that let WebSphere MQ File Transfer Edition exchange files based on common high-level qualifiers in the file names.

On Windows or UNIX, however, access is based on file ownership, directory ownership, and group affiliation. Some mapping of ownership and privileges is therefore necessary to get the fine-grained access control required.

Sandbox

WebSphere MQ File Transfer Edition provides a mechanism to restrict the area of a file system that an agent can access. To enable a sandbox, add the following property to the agent.properties file for the agent that you want to restrict:

```
sandboxRoot=<restricted_directory_name>
```

Using the sandbox might not eliminate the need for application-dedicated WebSphere MQ File Transfer Edition agents because that depends on the organization security requirements. It is recommended that you implement a security scheme that uses both application userIDs and a sandbox.

Working in a sandbox on z/OS

On z/OS, using a sandbox restricts the data set name qualifiers that the WebSphere MQ File Transfer Edition agent can read from and write to. The user that started the WebSphere MQ File Transfer Edition agent must have the correct operating system authorities to any data sets that are involved. If you enclose the sandboxRoot directory value in quotes, the value follows the normal z/OS convention and is treated as fully qualified. If you omit the quotes, 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 runtime, 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 distributed platforms

On distributed platforms, a sandbox restricts which directory a WebSphere MQ File Transfer Edition agent can read from and write to. When a sandbox is activated, the WebSphere MQ File Transfer Edition agent can read and write to the directory specified and only any subdirectories that the specified directory contains. A WebSphere MQ File Transfer Edition sandbox does not take precedence over operating system security. The user that started the WebSphere MQ 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 that it is linked to is outside of the specified sandboxRoot directory (and subdirectories).

Managing authorities to access file systems

To transfer a file, the userID that the source agent runs under must have read access to the source file. Additionally, you might need to give the source agent *delete authority* depending on the source disposition attribute.

If the files that you want to transfer are not in a location that is accessible to the agent, you can use WebSphere MQ File Transfer Edition source-side user exits to move the file to an accessible location, where the agent does have read access.

The destination agent process 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.

If the files that you transfer are not in a location that is accessible to the application, you can use WebSphere MQ File Transfer Edition destination-side user exits to move the file to a location where the application does have file access.

12.3.2 WebSphere MQ File Transfer Edition command and shell security

WebSphere MQ File Transfer Edition can integrate its file transfer functions using scripts that Apache Ant runs. Ant accepts a script file that is coded in XML. Within the XML script are *verbs*, known as *Ant Tasks*, that represent the actions that the script will perform. One such Ant task is the call tasks that allow you to remotely call scripts and programs.

The *call task* allows you to send a call request to an agent. The agent processes this request by running a script or program and returning the outcome. Place the commands that you want to be able to call in the path specified by the agent property commandPath in the agent.properties file. Do not include path information with the call command. By default commandPath is empty so that the agent cannot call any commands.

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 can send commands to the agent. For this reason, by default, when you specify a commandPath, a sandbox is enabled so that all commandPath directories are automatically denied access for a transfer. You can set the sandboxRoot property to override this default behavior, but you are not recommended to do so because it effectively enables a client to transfer any command to the agent's system and call it.

12.4 WebSphere MQ File Transfer Edition additional considerations

In this section, we address forward attacks and the role of topologies in security control.

12.4.1 Forwarding attacks

Because the WebSphere MQ File Transfer Edition agents can both read and write end-user data it provides an opportunity for what is know as a *file forwarding attack*. In a topology where there are two WebSphere MQ File Transfer Edition network domains with a gateway agent between them, an attack could use the intermediate or gateway agent to move data.

The attacking agent would deliver a file to an intermediate agent and then cause that agent to relay the file to a third agent to which the attacker does not have direct access. Because agents communicate with each other using the same command queue as end users, you cannot prevent this by blocking access to the remote agent's command queue. What you can do, however, is configure the agents to be read-only or write-only by editing the agent.properties file. Setting the maxSourceTransfers property to zero causes the agent to refuse to transfer outbound files. Similarly, setting the maxDestinationTransfers property to zero causes the agent to refuse to accept inbound files. Although it is syntactically correct to set both of these values to zero, it is not recommended because it completely disables the agent from transferring any files. If two agents are deployed on a server, with one as a file sender and one as a file receiver, configuring them so that their sandbox directories do not overlap provides protection against a file relay attack.

12.4.2 The role of topology as a security control

We saw that the WebSphere MQ side of WebSphere MQ File Transfer Edition is secured like any other MQ application. Queue security is based on userIDs and groups. In a typical network, access to FTE agents occurs through remote connections and portions of the security policy are implemented through the physical connectivity of the messaging network. To the extent that different access roles are required, these are governed by userIDs in the MCAUSER attributes of the various channels. Thus far these architectural aspects suggested a topology that provides separate paths for interactive commands versus inter-agent commands. It also resulted in the partitioning of agents into send-only and receive-only pairs and a prerequisite that the queue managers involved were all hardened against administrative privilege escalation.

Many shops will not implement this level of security in their existing WebSphere MQ network, for example, if the agents were placed into a WebSphere MQ cluster where users were routinely granted access to the cluster transmit queue, any user can drive any WebSphere MQ File Transfer Edition agent. Creating small closed networks of WebSphere MQ File Transfer Edition agents and users might be a common deployment pattern. The WebSphere MQ File Transfer Edition queue managers can be created alongside existing queue managers so that new WebSphere MQ installations are kept to a minimum, but it is easier to secure a new, small, closed network than to retrofit security into a large existing network that supports many critical applications. If the amount of data that is being transferred is significant, this could also be a method of not impacting the performance of existing applications.

The security roles that we discussed thus far included administrators, end users, and agents. But a typical requirement might be to have several different classes of end users and a granular security model that enforces restrictions against these users accessing each other's files. We already saw that WebSphere MQ File Transfer Edition security is expressed largely through the physical topology of the network, and a small closed network is one way to provide separation of duties between administrators and end users. You can expand on that model to provide granularity among end-user roles by building multiple closed WebSphere MQ File Transfer Edition networks, for example, there might be one WebSphere MQ File Transfer Edition network for Personnel and another for Payroll.

If separate closed networks of FTE agents are required to support a granular security model, use overlapping sandboxes for files to be exchanged securely, for example, if Personnel needs to send a new Employee Master file to Payroll, you can accomplish this by having agents from both departments using different agent queue managers but with their sandboxes configured to point to the same directory. Although files in the sandbox are accessible to both agents, the agent command queues are on separate queue managers and accessible only to their respective departments.

12.4.3 Additional information

For more information about securing WebSphere MQ File Transfer Edition, refer to the following article by T.Rob Wyatt:

http://www.ibm.com/developerworks/websphere/library/techarticles/0902_wyatt/0902_wyatt.html

13

Advanced topics

In this chapter, we discuss several advanced topics. In the first section, we describe *file transfer exits* of important extended functions. Exits permit users to develop and invoke Java functions. We also discuss using high availability (HA) with WebSphere MQ File Transfer Edition. HA can avoid a single point-of-failure on file transfer systems. In the final section, we cover performance considerations of WebSphere MQ File Transfer Edition. After reading this chapter, you will understand how to design reliable and efficient file transfer networks.

This chapter covers the following topics:

13.1 Using file transfer exits

Using WebSphere MQ File Transfer Edition users can extend the functionality of FTE by invoking *user-written Java code*. User written Java code is invoked at specific exit points in the life cycle of a transfer. So you can customize the features of WebSphere MQ File Transfer Edition by using your own Java programs, known as user exit routines.

13.1.1 Overview of FTE exits

WebSphere MQ File Transfer Edition provides points in the code where WebSphere MQ File Transfer Edition can pass control to a program that you developed. These points are known as *user exit points*. FTE can then resume control when your Java program completes its work. You do not have to use any of the user exits, but they are very useful if you want to extend the functions of file transfers to meet special requirements.

There are two points during file transfer processing where you can invoke a user exit at the source system, and there are two points during file transfer processing where you can invoke a user exit at the destination system. Here is a summary of each of these user exit points and their order:

- On the Source Agent:
  - Before the transfer starts
  - After the transfer completes
- On the Destination Agent:
  - Before the transfer starts
  - After the transfer completes

User exits are invoked in the following order:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Every exit is specified to a user exit Java class, as shown in Table 13-1.

Table 13-1 FTE user exits mapping to Java interfaces

Exit point	**Java interface**
SourceTransferStartExit	SourceTransferStartExit.java
DestinationTransferStartExit	DestinationTransferStartExit.java

Exit point	Java interface
DestinationTransferEndExit	DestinationTransferEndExit.java
SourceTransferEndExit	SourceTransferEndExit.java

For each exit point, you can specify multiple user exit Java classes. The order of execution of the user exits for a particular user exit point is determined by the order that the Java classes appear in the comma separated lists in the agent properties file. Any changes made to transfer metadata or file specifications are passed on to later user exits in the list.

The source start exits can update the list of source and destination files to be transferred. The destination user start exits can change the names of the destination file but cannot add or delete files. All user exits can return a result string that is then included in the transfer log. The user source and destination start exits return a result code that indicates whether to allow the transfer to continue or whether to cancel it.

13.1.2 Configuring the user exits

To configure user exits:

1. Set the agent.properties file.
2. Copy your JAR file to the agent's "exits" directory.

Setting the agent.properties file

You can set the agent.properties file by specifying user exit class names. There is one property for each user exit point. Each property is a comma separated list of class names. For an exit point, the user exits are executed in the order in which they appear in the comma separated list. Table 13-2 contains user exits' properties descriptions.

Table 13-2 User exits properties

Property name	Default value	Description
sourceTransferStartExitClasses	No default	Specifies a comma-separated list of classes that implement a source transfer start exit routine.
sourceTransferEndExitClasses	No default	Specifies a comma-separated list of classes that implement a source transfer end exit routine.

Property name	Default value	Description
destinationTransferStartExitClasses	No default	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.
destinationTransferEndExitClasses	No default	Specifies a comma-separated list of classes that implement a destination transfer end user exit routine.
exitClassPath	Agent's exits directory	Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines. The agent's exit directory is searched before any entries in this class path.
exitNativeLibraryPath	Agent's exits directory	Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.

The agent.properties file is located in the directory:
<WMQFTE_install_directory>/data/<qmgr_name>/agents/<agent_name>

Example: on Windows, the file location is located in
C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT

We describe one example of a user exits configuration in the agent.properties in Example 13-1.

Example 13-1 User exit routine properties

```
destinationTransferEndExitClasses=com.ibm.wmqfte.userexits.DestinationExits
sourceTransferStartExitClasses=com.ibm.wmqfte.userexits.SourceExits
destinationTransferStartExitClasses=com.ibm.wmqfte.userexits.DestinationExits
sourceTransferEndExitClasses=com.ibm.wmqfte.userexits.SourceExits
```

Copying your JAR file to the agent’s exits directory

Your user exits JAR files can be copied to the exits sub-directory of the agent’s directory:
<WMQFTE_install_directory>/data/<qmgr_name>/agents/<agent_name>/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.

13.1.3 Metadata for user exit routines

The user can specify user transfer metadata, which is then passed to the user exits. You can specify this using the `fteCreateTransfer` command or using the WebSphere MQ Explorer New Transfer wizard. The source user start exits can update the user transfer metadata.

There are three different types of metadata that can be supplied to user exit routines for WebSphere MQ File Transfer Edition: 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.

Transfer metadata

Transfer metadata is passed to all user exit routines. The initial values for the source transfer start user exit are based on those that you supply when you define the transfer. The source agent can change values as part of the processing of the source transfer start user exit. This user exit is called before the 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. You must define your own key-value pairs for your user exits: WebSphere MQ File Transfer Edition cannot create these definitions.

The following information cannot be modified by the user exits:

- MQMD User
- Originating User
- Transfer id
- Originating Host
- Source Agent name
- Destination Agent name
- JobName

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.

More information about metadata is at:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/metadata.htm

13.1.4 Java interfaces of FTE user exits

FTE provides four standard Java user exit interface programs. You can directly use this program structure for your functions.

SourceTransferStartExit.java

The SourceTransferStartExit.java source transfer start interface, Example 13-2, defines a method that is invoked immediately prior to starting a transfer on the agent that is acting as the source of the transfer.

Example 13-2 SourceTransferStartExit.java

```
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

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 <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 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 <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 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);
}
```

SourceTransferEndExit.java

The SourceTransferEndExit.java source transfer end interface, shown in Example 13-3, defines a method that is immediately invoked after completing a transfer on the agent that is acting as the source of the transfer.

Example 13-3 SourceTransferEndExit.java

```
package com.ibm.wmqfte.exitpoint.api;

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 <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 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 onSourceTransferEnd(TransferExitResult transferExitResult,
              String sourceAgentName,
              String destinationAgentName,
              Map<String, String>environmentMetaData,
              Map<String, String>transferMetaData,
              List<FileTransferResult>fileResults);
}
```

DestinationTransferStartExit.java

The DestinationTransferStartExit.java destination transfer start interface, shown in Example 13-4, defines a method that is called by the destination agent prior to the file being transferred.

Example 13-4 DestinationTransferStartExit.java

```
package com.ibm.wmqfte.exitpoint.api;

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);
}
```

DestinationTransferEndExit.java

The DestinationTransferEndExit.java destination transfer end interface, shown in Example 13-5, defines a method that is called by the destination agent after the entire file transfer is complete.

Example 13-5 DestinationTransferEndExit.java

```
package com.ibm.wmqfte.exitpoint.api;

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 <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 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);
}
```

13.1.5 Exits scenario

In this section, we provide an example for building a user exit.

JKHL scenario

JKHL submits one new requirement for file transfers from Washington D.C. to New York. Only the transaction ID is attached in the transferred document as metadata. This attachment format is location name (agent name) in conjunction with the transaction id by asterisk (*), for example, if the document is transferred from Washington to New York, the transaction ID is WASH*1234567 on the source agent in Washington and the transaction ID becomes NY*1234567 on the destination agent in New York.

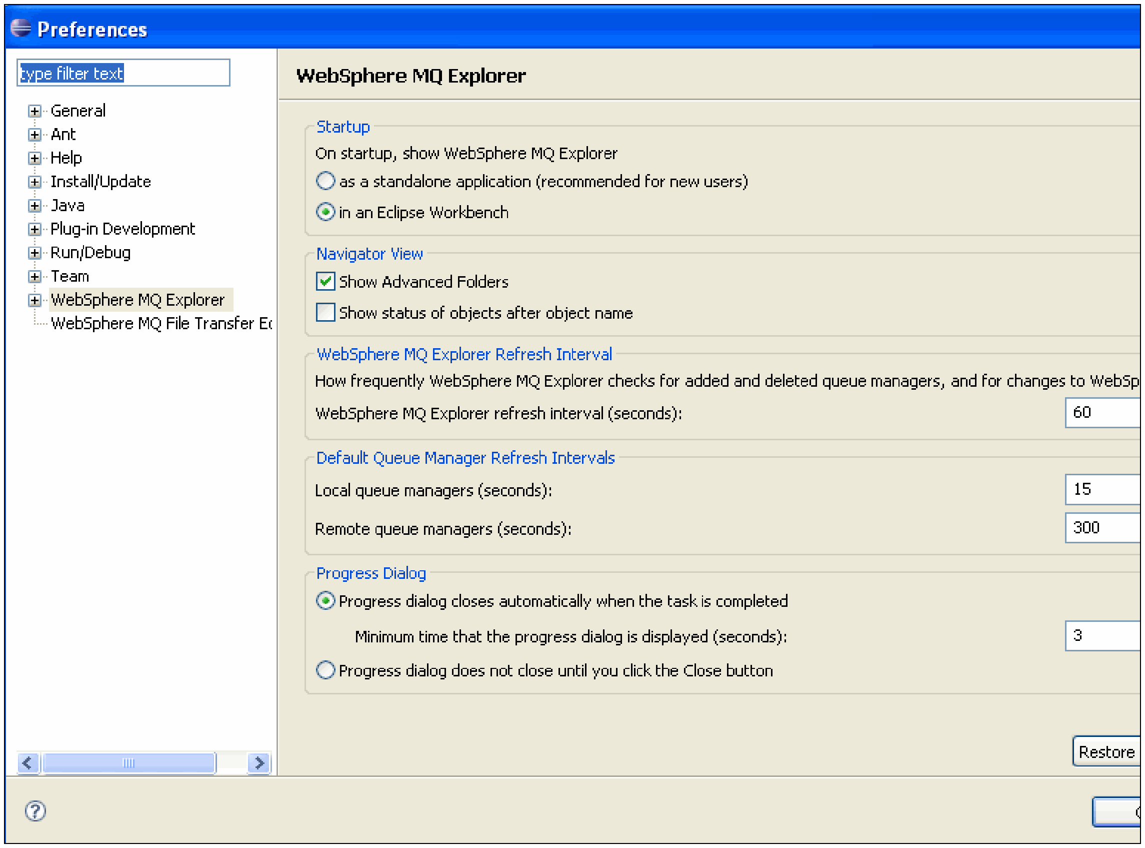
We can use FTE exits to implement this function by using a Java interface. The interface program captures the source transaction ID and the destination agent name when the file transfer starts. Then, the destination ID is created using only the transaction ID and destination agent name.

In this example, the Washington agent (WASH.AGENT) is running on WASHQM for Windows and the New York agent (NY.AGENT) is running on NYQM for Linux.

Java interface programming

You can use WebSphere MQ Explorer to develop Java code on WebSphere MQ Version 7.0. This new function is useful because you can manage WebSphere MQ objects and develop WebSphere MQ Java applications from the same interface. If you want to use a Java perspective:

1. Open WebSphere MQ Explorer, and select **Windows** → **Preferences** → **WebSphere MQ Explorer**.
2. In the Startup options, choose **in an Eclipse Workbench**, as shown in Figure 13-1.



WebSphere MQ Explorer

Startup

On startup, show WebSphere MQ Explorer

( ) as a standalone application (recommended for new users)

(●) in an Eclipse Workbench

Navigator View

☑ Show Advanced Folders

☐ Show status of objects after object name

WebSphere MQ Explorer Refresh Interval

How frequently WebSphere MQ Explorer checks for added and deleted queue managers, and for changes to WebSp

WebSphere MQ Explorer refresh interval (seconds): 60

Default Queue Manager Refresh Intervals

Local queue managers (seconds): 15

Remote queue managers (seconds): 300

Progress Dialog

(●) Progress dialog closes automatically when the task is completed

Minimum time that the progress dialog is displayed (seconds): 3

( ) Progress dialog does not close until you click the Close button

Figure 13-1 WebSphere MQ Explorer setting

3. Click **OK**, and restart WebSphere MQ Explorer.

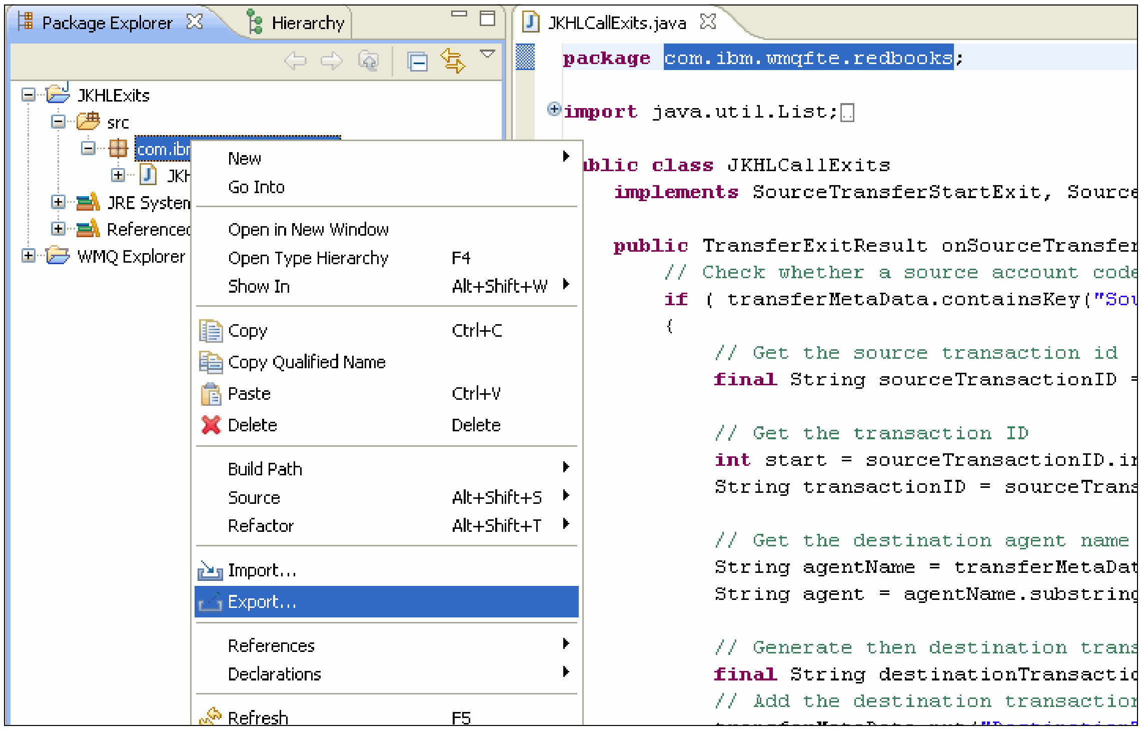
Now you can edit Java code in the Java perspective in WebSphere MQ Explorer.

Example 13-6 is a Java exit program written in WebSphere MQ File Transfer Edition.

Example 13-6 JKHLCallExits.java

```
package com.ibm.wmqfte.redbooks;

import java.util.List;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.DestinationTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.DestinationTransferStartExit;
import com.ibm.wmqfte.exitroutine.api.FileTransferResult;
import com.ibm.wmqfte.exitroutine.api.Reference;
import com.ibm.wmqfte.exitroutine.api.SourceFileExitFileSpecification;
import com.ibm.wmqfte.exitroutine.api.SourceTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.SourceTransferStartExit;
import com.ibm.wmqfte.exitroutine.api.TransferExitResult;
import com.ibm.wmqfte.exitroutine.api.TransferExitResultCode;

public class JKHLCallExits implements SourceTransferStartExit, SourceTransferEndExit,
        DestinationTransferStartExit, DestinationTransferEndExit {

   public TransferExitResult onSourceTransferStart(String sourceAgentName,String
destinationAgentName, Map<String, String> environmentMetaData, Map<String, String>
transferMetaData, List<SourceFileExitFileSpecification> fileSpecs) {
      // Check whether a source account code exists
      if ( transferMetaData.containsKey("SourceTransactionID"))
      {
         // Get the source transaction id
         final String sourceTransactionID =
transferMetaData.get("SourceTransactionID");

         // Get the transaction ID
         int start = sourceTransactionID.indexOf("*");
         String transactionID = sourceTransactionID.substring(start + 1);

         // Get the destination agent name
         String agentName = transferMetaData.get("com.ibm.wmqfte.DestinationAgent");
         String agent = agentName.substring(0, agentName.indexOf("."));

         // Generate then destination transaction id
         final String destinationTransactionID = agent + "*" + transactionID;
         // Add the destination transaction id to the transfer meta data
         transferMetaData.put("DestinationTransactionID", destinationTransactionID);
         // Pass back the transaction id and tell the transfer to Proceed
         return new TransferExitResult(TransferExitResultCode.PROCEED, "Source 
Transaction ID: " + sourceTransactionID);
      }
      else
      {
         // There is no source transaction id specified so cancel the transfer
         return new TransferExitResult(TransferExitResultCode.CANCEL_TRANSFER, "No 
Source Transaction ID");
      }
   }

   public String onSourceTransferEnd(TransferExitResult transferExitResult, String 
sourceAgentName, String destinationAgentName, Map<String, String> 
environmentMetaData, Map<String, String> transferMetaData, List<FileTransferResult> 
fileResults) {
      if ( transferMetaData.containsKey("SourceTransactionID"))
      {
         return "Source End. Transaction ID: " + 
transferMetaData.get("SourceTransactionID");
      }
      else
      {
         return "Source end";
      }
   }

   public TransferExitResult onDestinationTransferStart(String sourceAgentName, 
String destinationAgentName, Map<String, String> environmentMetaData, Map<String, 
String> transferMetaData, List<Reference<String>> fileSpecs) {
      // Check whether a destination transaction id exists
      if ( transferMetaData.containsKey("DestinationTransactionID"))
      {
         // Get the destination transaction id
         final String destinationTransactionID = 
transferMetaData.get("DestinationTransactionID");
         // If the destination transaction id is longer than 8 characters it is 
invalid
         if ( destinationTransactionID.length() >= 8)
         {
            // Pass back the transaction id and tell the transfer to Proceed
            return new TransferExitResult(TransferExitResultCode.PROCEED, "Destination 
Transaction ID: " + destinationTransactionID);
         }
         else
         {
            // The destination transaction id is invalid so pass back and cancel the
transfer
            return new TransferExitResult(TransferExitResultCode.CANCEL_TRANSFER,
"Destination Transaction ID Invalid: " + destinationTransactionID);
         }
      }
      else
      {
         // There is no destination transaction id specified so cancel the transfer
         return new TransferExitResult(TransferExitResultCode.CANCEL_TRANSFER, "No
Destination Transaction ID");
      }
   }

   public String onDestinationTransferEnd(TransferExitResult transferExitResult,
String sourceAgentName, String destinationAgentName, Map<String, String>
environmentMetaData, Map<String, String> transferMetaData, List<FileTransferResult>
fileResults) {
      if ( transferMetaData.containsKey("DestinationTransactionID"))
      {
         return "Destination End. Transaction ID: " +
transferMetaData.get("DestinationTransactionID");
      }
      else
      {
         return "Destination end";
      }
   }
}
```

You can export the Java class as a JAR file from the Java perspective in WebSphere MQ Explorer. We save the Java class to JKHLexits.jar, as shown in Figure 13-2 on page 330.



```
package com.ibm.wmqfte.redbooks;

import java.util.List;

ublic class JKHLCallExits
    implements SourceTransferStartExit, Source

    public TransferExitResult onSourceTransfe
        // Check whether a source account code
        if ( transferMetaData.containsKey("Sou
        {
            // Get the source transaction id
            final String sourceTransactionID =

            // Get the transaction ID
            int start = sourceTransactionID.i
            String transactionID = sourceTrans

            // Get the destination agent name
            String agentName = transferMetaDat
            String agent = agentName.substring

            // Generate then destination trans
            final String destinationTransacti
            // Add the destination transaction
```

Figure 13-2 Exporting Java class to jar file in WebSphere MQ Explorer

Configuring user exits

To run the file transfer with exits, you must configure the agent.properties file and copy the JAR file to the agent exits directory:

1. Edit the agent.properties file on WASH.AGENT for Windows, and add the contents in Example 13-7.

Example 13-7 Editing Washington agent's property file for Windows

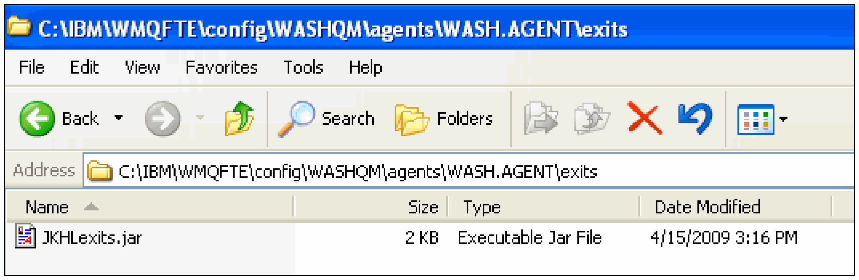
```
sourceTransferStartExitClasses=com.ibm.wmqfte.redbooks.JKHLCallExits
sourceTransferEndExitClasses=com.ibm.wmqfte.redbooks.JKHLCallExits
```

2. Edit agent.properties on NY.AGENT for Linux, and add the contents in Example 13-8.

Example 13-8 Editing NewYork agent's property file for Linux

```
destinationTransferStartExitClasses=com.ibm.wmqfte.redbooks.JKHLCallExits
destinationTransferEndExitClasses=com.ibm.wmqfte.redbooks.JKHLCallExits
```

3. Copy the jar file to the agent exits directory, as shown in Figure 13-3.



Address: C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT\exits

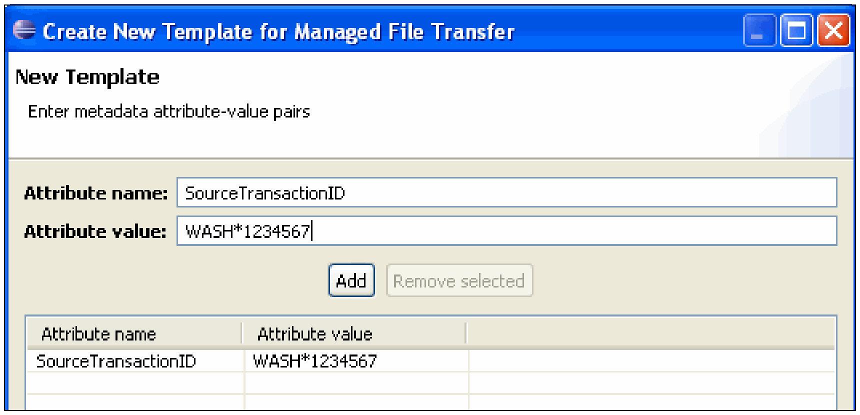
Name	Size	Type	Date Modified
JKHLexits.jar	2 KB	Executable Jar File	4/15/2009 3:16 PM

Figure 13-3 Exits jar file is put in the Agent exits directory

After you complete these steps, you can test the file transfer with exits.

Running the exits

In WebSphere MQ Explorer, to use the WebSphere MQ File Transfer Edition plug-in, you must create a new transfer, and enter agent and transfer information. When you enter the metadata windows, you input the information, as shown in Figure 13-4. Finally, you submit the file transfer.


Create New Template for Managed File Transfer
New Template
Enter metadata attribute-value pairs
Add
Remove selected

Attribute name: SourceTransactionID

Attribute value: WASH*1234567

Attribute name	Attribute value
SourceTransactionID	WASH*1234567

Figure 13-4 Setting file transfer metadata

If you issued the command with exits parameters, the command is similar to Example 13-9 on page 331.

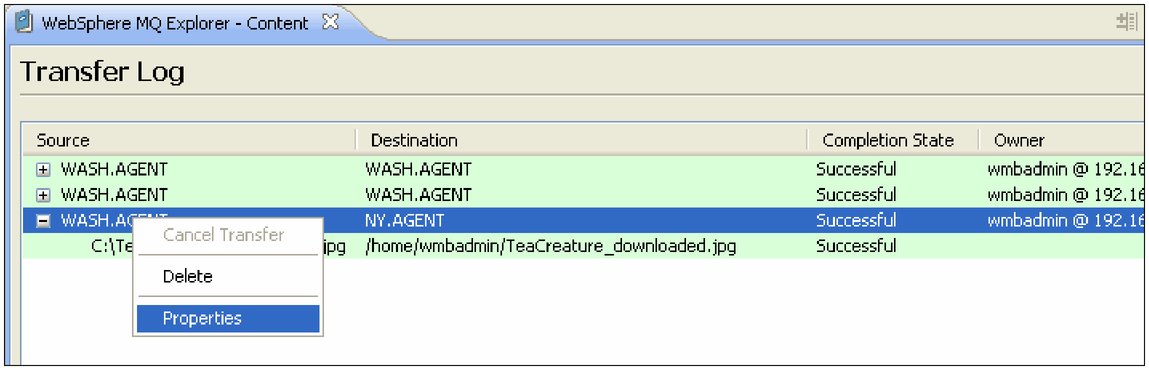
Example 13-9 File transfer command with exits

```
fteCreateTransfer -sa WASH.AGENT -sm WASHQM -da NY.AGENT -dm NYQM -de
overwrite -md SourceTransactionID=WASH*1234567 -df
/home/wmbadmin/WMQFTE.jpg C:\WMQFTE.jpg
```

Viewing the issued result

You can look through the transfer log for exits information:

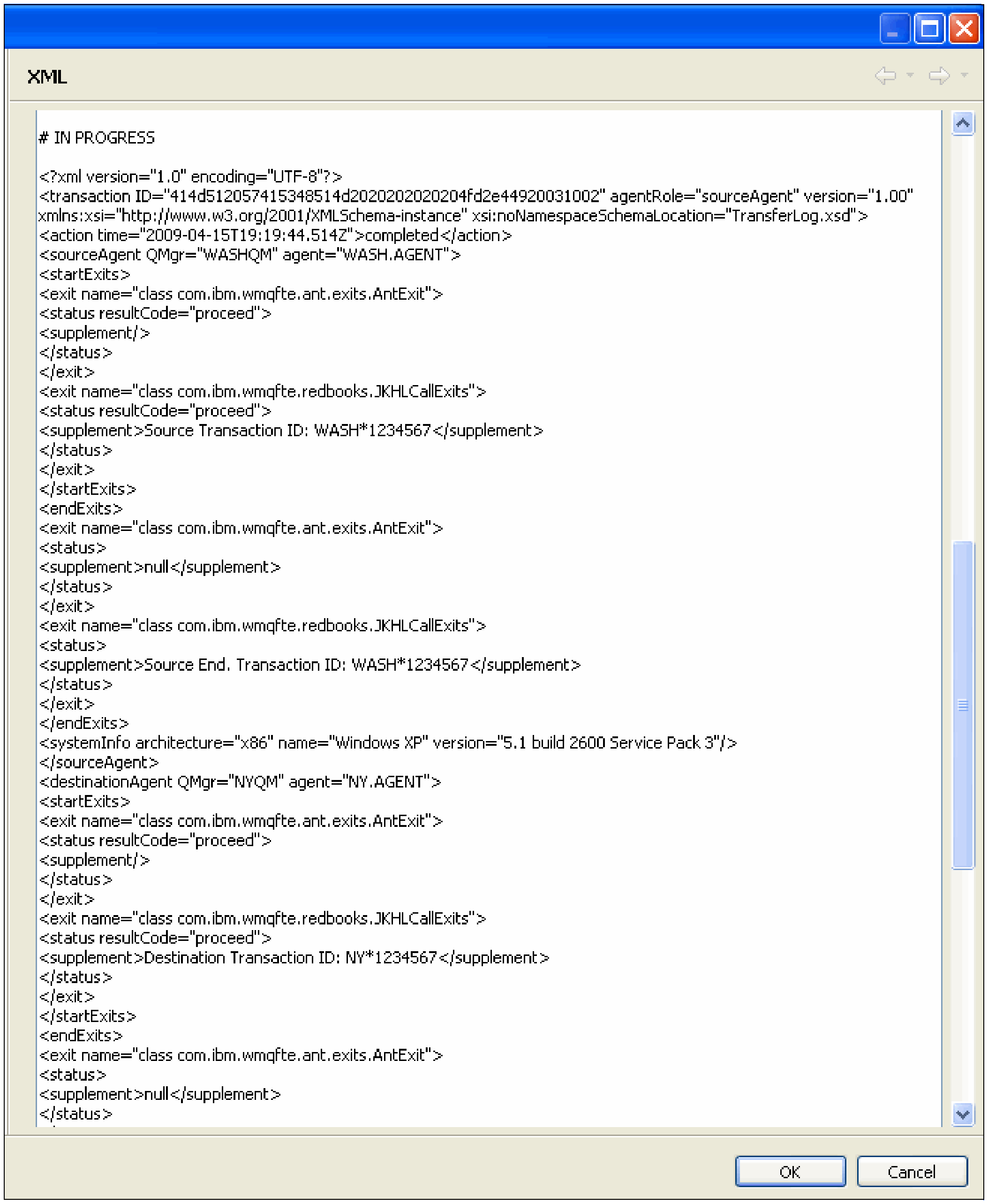
1. Right-click the item in the transfer log, and select **Properties**, as shown in Figure 13-5.



Source	Destination	Completion State	Owner
WASH.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.16
WASH.AGENT	WASH.AGENT	Successful	wmbadmin @ 192.16
WASH.A	NY.AGENT	Successful	wmbadmin @ 192.16
C:\Te jpg	/home/wmbadmin/TeaCreature_downloaded.jpg	Successful	

Figure 13-5 Selecting the transfer log properties

Figure 13-6 on page 333 shows the exits and metadata information.



```
# IN PROGRESS

<?xml version="1.0" encoding="UTF-8"?>
<transaction ID="414d512057415348514d2020202020204fd2e44920031002" agentRole="sourceAgent" version="1.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="TransferLog.xsd">
<action time="2009-04-15T19:19:44.514Z">completed</action>
<sourceAgent QMgr="WASHQM" agent="WASH.AGENT">
<startExits>
<exit name="class com.ibm.wmqfte.ant.exits.AntExit">
<status resultCode="proceed">
<supplement/>
</status>
</exit>
<exit name="class com.ibm.wmqfte.redbooks.JKHLCallExits">
<status resultCode="proceed">
<supplement>Source Transaction ID: WASH*1234567</supplement>
</status>
</exit>
</startExits>
<endExits>
<exit name="class com.ibm.wmqfte.ant.exits.AntExit">
<status>
<supplement>null</supplement>
</status>
</exit>
<exit name="class com.ibm.wmqfte.redbooks.JKHLCallExits">
<status>
<supplement>Source End. Transaction ID: WASH*1234567</supplement>
</status>
</exit>
</endExits>
<systemInfo architecture="x86" name="Windows XP" version="5.1 build 2600 Service Pack 3"/>
</sourceAgent>
<destinationAgent QMgr="NYQM" agent="NY.AGENT">
<startExits>
<exit name="class com.ibm.wmqfte.ant.exits.AntExit">
<status resultCode="proceed">
<supplement/>
</status>
</exit>
<exit name="class com.ibm.wmqfte.redbooks.JKHLCallExits">
<status resultCode="proceed">
<supplement>Destination Transaction ID: NY*1234567</supplement>
</status>
</exit>
</startExits>
<endExits>
<exit name="class com.ibm.wmqfte.ant.exits.AntExit">
<status>
<supplement>null</supplement>
</status>
```

Figure 13-6 Viewing the file transfer log with exits information

13.2 High availability of WebSphere MQ File Transfer Edition

In this section, we discuss several methods of implementing high availability in WebSphere MQ File Transfer Edition. FTE is a pure WebSphere MQ Java application, so FTE HA is fully based on WebSphere MQ HA for open platforms including Windows, UNIX, and Linux.

In this section, the situation that we describe about HA is that one agent and its queue manager reside on the same physical OS and machine. The agent can use binding or client mode to connect to its queue manager in the same hardware environment.

The common HA method for WebSphere MQ that we discuss is a hardware cluster that supports an active-standby setup on queue managers. In the following sections, we focus on the hardware cluster HA configuration in WebSphere MQ File Transfer Edition.

13.2.1 Using active-standby without shared disks

A full HA clustering is not always necessary when considering an HA solution. If you are willing to resend a file transfer request at a later time when the master queue manager or server is out of service, you can use the simple HA method for FTE without shared disks. If you do not care about transfer log loss on the coordination queue manager, you can also use this solution. In Figure 13-7 on page 335 there is an FTE master server (active machine) and secondary server (standby machine) in Washington, and an HA software is installed and configured in the two servers.

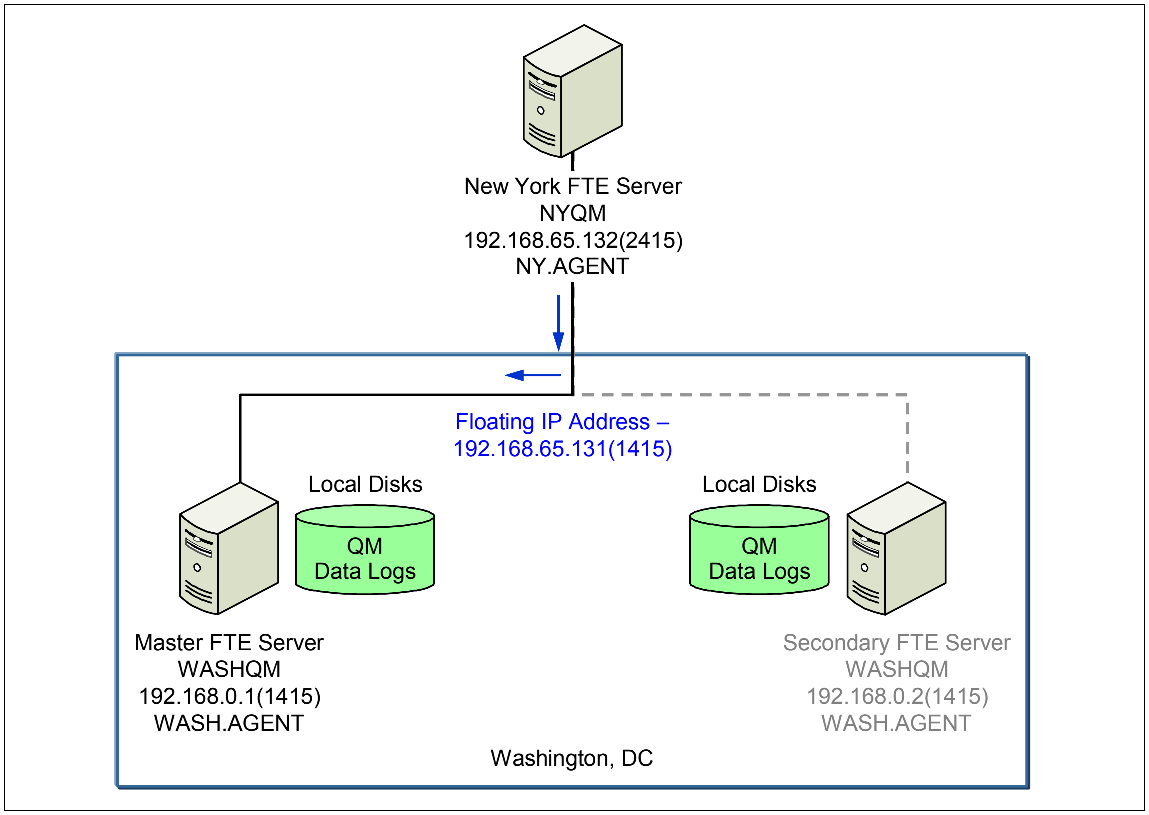


Figure 13-7 Active-Standby mode without shared disks

FTE agents and queue managers in two machines are set to the same configuration, including agent name, QM name, queue name, and so on. They use internal IP addresses (192.168.0.1 and 192.168.0.2) in the hardware cluster and share the same floating IP address (192.168.65.131) that the HA software manages. In the default configuration, the floating IP is directed to the FTE master server.

Two servers both utilize the local disks for logs and queues. You can still set FTE system queues (SYSTEM.FTE, SYSTEM.FTE.COMMAND.<AGENT>, SYSTEM.FTE.STATE.<AGENT>, and SYSTEM.FTE.EVENT.<AGENT> queues) to persistence for transfer log integrity, agent recovery, and file transfer checkpoint. This set up means that if agent recoverable exceptions and network timeouts occur, file transfer commands and statuses can be kept in the persistent system queues that reside on local disks, the file transfer is recovered to the last checkpoint when the agent begins to work, and the network is connected between source and destination.

Under normal circumstances, the files that are transferred from the FTE server on New York are sent to the master FTE server on Washington. When the master FTE queue manager on Washington is not available, the HA software can find this failure within the heartbeat interval and switch the floating IP to a standby FTE server. When the new file transfer request is submitted, a secondary FTE server starts to take over and transfers files.

We suggest that before you want to recover the master machine, you kill the agent, do an **`fteCleanAgent`** on the agent in the master machine, and then restart the agent again. When the agent is restarted with empty FTE queues, it might be necessary to clean up the transfers from the other agents that are participating in the transfers that were current when the agent failed over to the other server. Use the **`fteCancelTransfer`** command to do this if the other agents and the transfer IDs are known. If they are not cleaned details, the original transfers will remain in the SYSTEM.FTE.STATE.<AgentName> queue of the agents. Also, the SYSTEM.FTE.EVENT.<AgentName> queue holds details of scheduled transfers and directory monitors, and these are not copied across to the new server.

Note: In Figure 13-7, if WASHQM is the coordination queue manager and the log subscribers use the non-persistent mode, the transfer logs are partially lost during the process of loading logs into the database when the queue manager or server is out of work.

13.2.2 Using active-standby with shared disks

In the standby solution, the secondary server is used to host a second queue manager that is activated when the master server or queue manager fails. The standby server also needs to be an exact replica, at any broken point-in-time of the master machine, so that when failure occurs, the standby machine can start file transfers correctly from the last checkpoint. That is, the FTE queue manager and agent on the standby server should be at the same configuration as the master FTE server, and the standby machine must have the same security privileges that the primary server has.

In Figure 13-8 on page 337, FTE agents and QMs in two machines are set to the same configuration, including agent name, QM name, queue name, and so on. They use internal IPs (192.168.0.1 and 192.168.0.2) in a hardware cluster and share the same floating IP address (192.168.65.131) that is managed by HA software. In the default configuration, the floating IP is directed to the FTE master server.

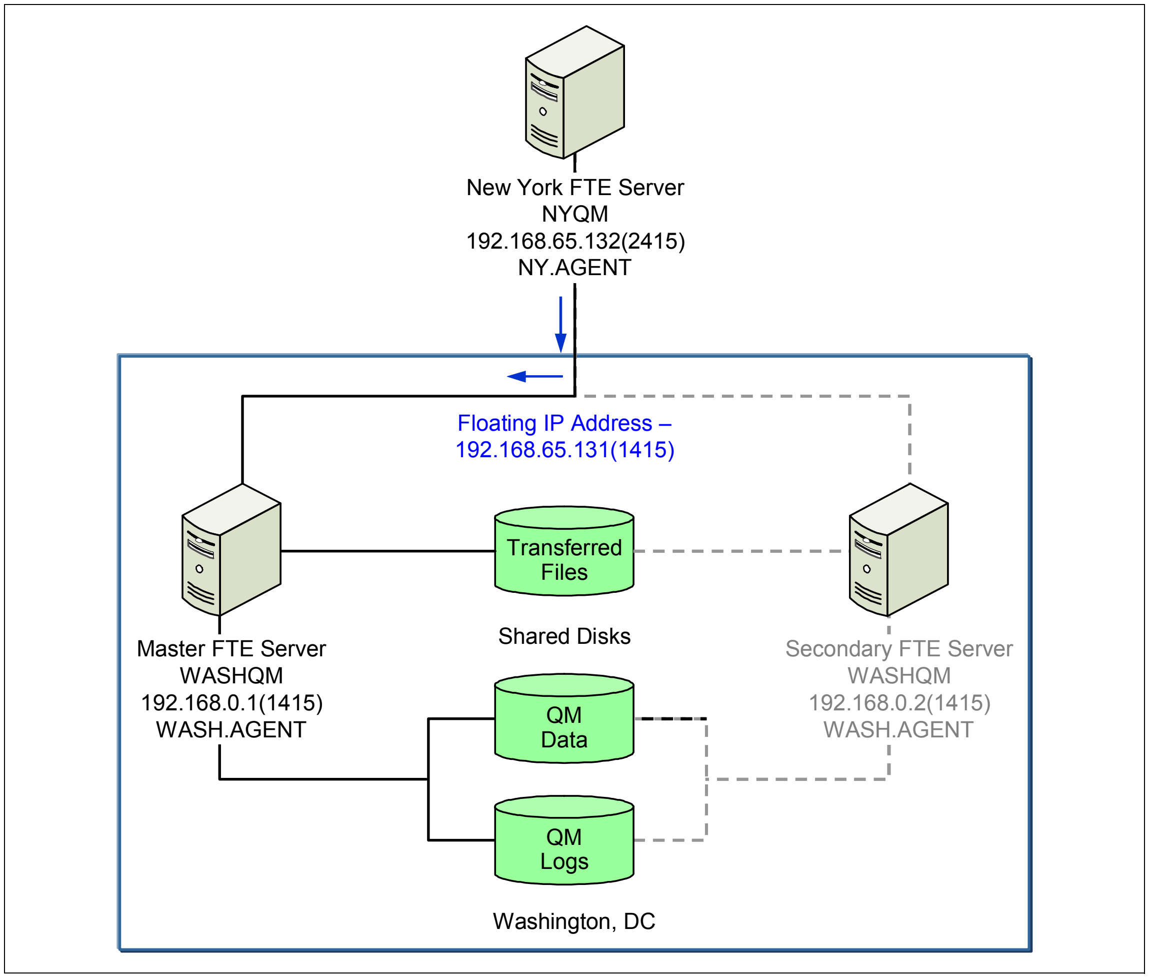


Figure 13-8 Active-standby mode with shared disks

The difference in this solution from the previous solution is that a shared disk device is used to provide a resilient store for processed files, queue data, and queue manager logs so that file redundancy transfer is avoided. This process ensures that files can continue to be transferred from the checkpoint when the standby server starts. In most scenarios, customers do not expect this file to transfer from the start point again after the queue manager or server recovers. If file transfer data, including command, status and other event information, can be replicated to a standby FTE server using shared disks, it assumes that when the master FTE queue manager or server fails, the standby server can take over the master server's role and continue processing file transfers from the checkpoint information that is stored on the shared disks.

The following WebSphere MQ File Transfer Edition objects are stored in shared disks:

- Four persistent queues on QM:
  - SYSTEM.FTE
  - SYSTEM.FTE.COMMAND.<AGENT>
  - SYSTEM.FTE.STATE.<AGENT>
  - SYSTEM.FTE.EVENT.<AGENT>
- MQ logs
- Transferred files

13.3 Performance considerations

WebSphere File Transfer Edition provides powerful and efficient functions to transfer large files that are more than 1 GB and thousand of files whose sizes are less than 100 KB. We must consider the different performance factors according to some real scenarios.

WebSphere MQ File Transfer Edition must first be tuned on the following parameters and properties of WebSphere MQ queue managers: queues, channels, listeners, and logs. For multi-threaded transfers, we can modify the FTE agent properties to increase the transfer performance. We set the optimal operation system parameters for file transfers because WebSphere MQ File Transfer Edition depends on an OS environment. In the following sections, we discuss these advanced topics.

13.3.1 WebSphere MQ performance considerations

For WebSphere MQ File Transfer Edition:

- The data and acknowledgement messages are put non-persistently to the SYSTEM.FTE.DATA.<AgentName> and SYSTEM.FTE.REPLY.<AgentName> queues.
- Messages are put persistently to the SYSTEM.FTE.COMMAND.<AgentName>, SYSTEM.FTE.EVENT.<AgentName>, and SYSTEM.FTE.STATE.<AgentName> queues.

But the number of persistent messages is small compared to the non-persistent messages. So the precondition we discuss in this section is based on WebSphere MQ non-persistent messages.

Non-persistent messages

Non-persistent messages are held in the main memory and are cached to the file system when the queues become deep. Then a few messages are lazily written to the queue file. It is an advantage that non-persistent messages do not need to be written into WebSphere MQ logs for recovery. This mechanism decreases file system I/O, but there is still file system I/O with non-persistent messages. When the queue buffer size or system memory is not adequate, the non-persistent messages will still cause file system I/O.

Non-persistent queue buffer size

There are two buffers to hold messages in WebSphere MQ queues. One buffer is for non-persistent messages and the other is for persistent messages. After non-persistent messages overflow from the system memory, they are moved to a file system on the operating system. You can modify the non-persistent queue buffer size to limit this overspill so that messages are processed quickly on the queue manager. The default non-persistent queue buffer size is 64 K for the 32-bit queue managers (Windows32, Linux32) and 128 K for 64-bit queue managers (AIX, Solaris, HPUX, Linux64, zOS, and Windows64). The maximum supported size on WebSphere MQ is 100 MB. You can define the parameter DefaultQBufferSize as the non-persistent queue buffer size. The value is taken from the TuningParameters stanza that is in use by the queue manager.

Note: Defining queues using large non-persistent or persistent queue buffers can degrade performance if the system is short of real memory either because a large number of queues were already defined with large buffers, or for other reasons, for example large number of channels defined.

You must consider the appropriate buffer value that comes from the real throughput and the average message size on the queue.

If you want to set the queue buffer size for FTE system queues, you must change the *DefaultQBufferSize* value in the queue manager qm.ini file, as shown in Example 13-10.

Example 13-10 Setting the parameter DefaultQBufferSize in qm.ini

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : WebSphere MQ queue manager configuration file      *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                *#
#******************************************************************#
TuningParameters:
   DefaultQBufferSize=1024000
```

Channels

We describe the configuration of WebSphere MQ channels to optimize performance. The channel tuning parameters are:

- MQIBindType in qm.ini
- PipeLineLength in qm.ini
- Compression on Channel property
- NPMSPEED on Channel property
- BATCHSIZE

An MQ channel listener supports two kinds of operation modes: trusted (fastpath) and non trusted (standard). The setting of trusted mode can reduce CPU and memory consumption. You can set the value of parameter *MQIBindType* using WebSphere MQ Explorer on Windows and Linux (32-bit).

If you want to change a parameter value on Linux and UNIX, you can set the MQIBindType value in the queue manager qm.ini file, as shown in Example 13-11.

Example 13-11 Setting the parameter MQIBindType in qm.ini

```
#******************************************************************#
#* Module Name: qm.ini                                            *#
#* Type       : WebSphere MQ queue manager configuration file     *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                *#
#******************************************************************#
#* Notes      :                                                   *#
#* 1) This file defines the configuration of the queue manager    *#
#*                                                                *#
#******************************************************************#
Channels:
   MQIBindType=FASTPATH
```

You can also optionally allow a *message channel agent* (MCA) to transfer messages using multiple threads. This process, called *pipelining*, enables the MCA to transfer messages more efficiently, with fewer wait states, which improves channel performance. Each MCA is limited to a maximum of two threads.

With WebSphere MQ for Windows and Linux (32-bit), you can use the WebSphere MQ Explorer to set the *PipeLineLength* parameter in the Channels properties of the queue manager.

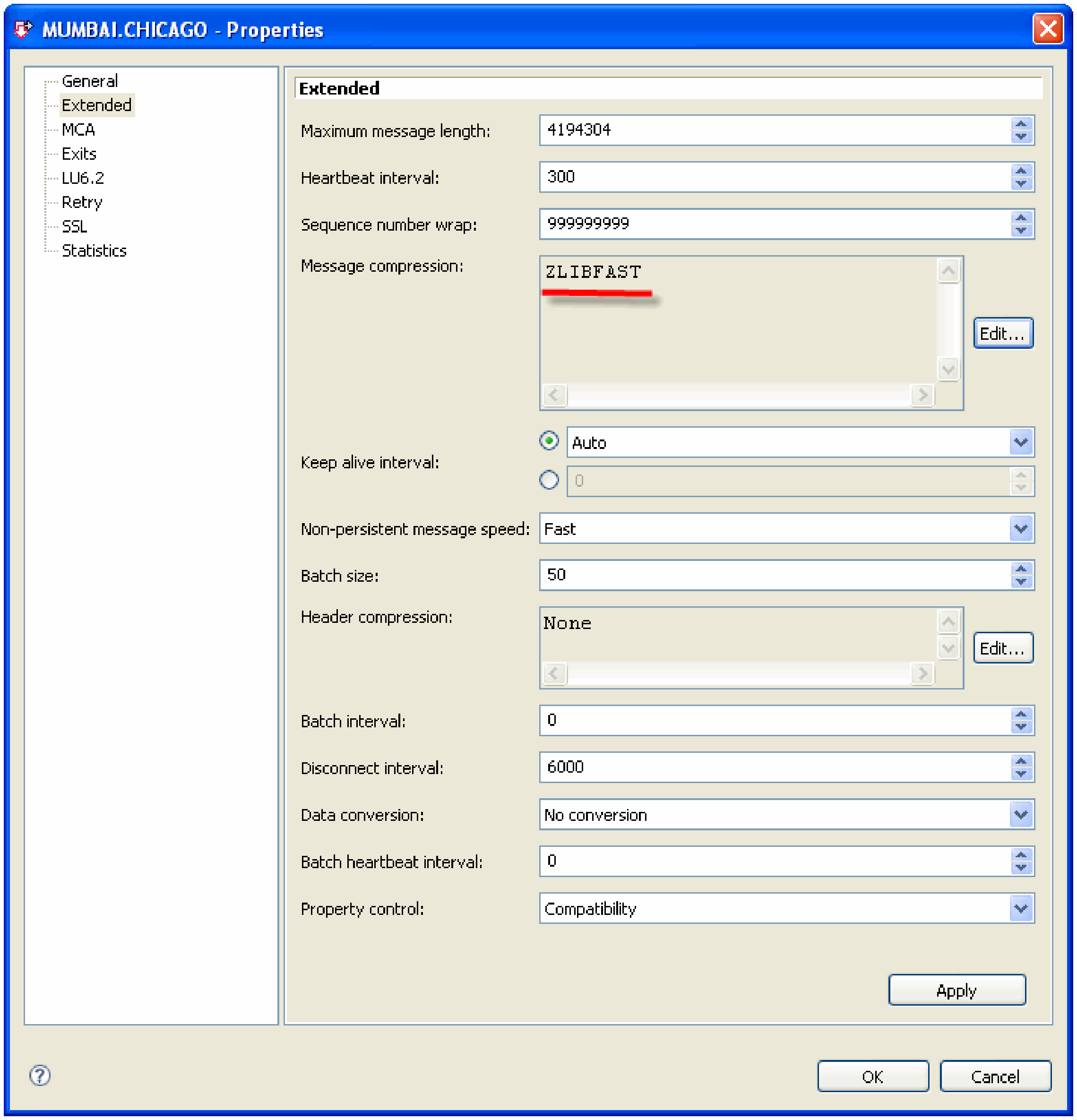
For UNIX and Linux, you control pipelining with the PipeLineLength parameter in the qm.ini file. This parameter is added to the CHANNELS stanza, as shown in Example 13-12.

Example 13-12 Setting the pipeline parameter

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : WebSphere MQ queue manager configuration file      *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
Channels:
   PipeLineLength=2
```

Transferring large-size files can be a strain on the network. We must consider how to decrease the network bandwidth usage. Compressing the data that flows on a WebSphere MQ channel can improve the performance of the channel and reduce network traffic. Using functions that are supplied with WebSphere MQ, you can compress the data that flows on message channels and MQI channels, and on either types of channels, you can compress header data and message data independently. But there are two types of files for transfer. One type is a pure text file that can be compressed at high rates, which at its peak is above 90 percent. The other is media or picture files that cannot be compressed well. If you set the compression parameter for these types of files, it is not useful.

By default, no data or message is compressed on a channel. You must change the compression parameter on both sender and receiver channels in WebSphere MQ Explorer. You can select an appropriate compression algorithm from either RLE, ZLIBFAST and ZLIBHIGH. See Figure 13-9 on page 342. You can also add algorithms from external applications.


MUMBAI.CHICAGO - Properties

- General
- Extended
- MCA
- Exits
- LU6.2
- Retry
- SSL
- Statistics

Extended

Maximum message length:	4194304
Heartbeat interval:	300
Sequence number wrap:	999999999
Message compression:	ZLIBFAST
Keep alive interval:	[✓ Auto] / 0
Non-persistent message speed:	Fast
Batch size:	50
Header compression:	None
Batch interval:	0
Disconnect interval:	6000
Data conversion:	No conversion
Batch heartbeat interval:	0
Property control:	Compatibility

Figure 13-9 Setting the channel compression

We can use the **runmqsc** WebSphere MQ administration command to change the message compression property on UNIX and Linux, as shown in Example 13-13 on page 343.

Example 13-13 Setting the channel compression

```
ALTER CHANNEL(ChannelName) CHLTYPE(SDR|RCVR)
   COMPMSG(RLE|ZLIBFAST|ZLIBHIGH)
```

The NPMSPEED parameter refers to non-persistent message speed. The default is FAST. The advantage of this is that non-persistent messages become available for retrieval far more quickly, which means that MQ Channels that are running with the FAST parameter value are higher performance than those on the NORMAL value. The value on Windows and Linux (32-bit) might be changed in WebSphere MQ Explorer.

The channel batch size parameter has a significant impact on channel performance. When WebSphere MQ transfers messages, MQ channels process those messages under the control of a synchronous point. Under the following conditions it submits a number of uncommitted messages:

- The number of messages reaches the value of batch size
- The transmission queue is empty and no new messages arrive within the time interval that the batch interval parameter specifies.

By default, the value of batch size on a channel is 50, which is a more reasonable and optimized setting. If the batch size value is set too small, every message that the channel processes will occupy a large amount of system resources. You can adjust batch size to adapt to different networks.

13.3.2 Performance tuning FTE agents

You can change an agent's parameters to increase FTE performance. If you want to modify the agent parameters, access the agent.properties file. The agent.properties file for an agent is located in your <WMQFTE_config_directory>/<coordination_qmgr_name>/agents/<agent_name> directory, for example, on Windows, the file location is located in C:\IBM\WMQFTE\config\WASHQM\agents\WASH.AGENT.

Table 13-3 lists the agent size properties.

Table 13-3 Agent size properties

Property name	Default value	Description
agentFrameSize	5	The number of windows for the transfer frame.
agentWindowSize	10	The number of chunks for each window.
agentChunkSize	256 KB	The size of each transfer chunk for the transport of file data, actual MQ message data size.

Property name	Default value	Description
agentCheckpointInterval	1	The interval in terms of complete frames of data at which a checkpoint is taken for recovery purposes. If a transfer fails, the transfer can only recover at checkpoint boundaries. Hence the size of this value (together with large-chunk window and frame values) increases the time that the agent takes to recover failed transfers.

Here is brief explanation of the concepts that are associated with the agent size parameters:

- The frame size determines the maximum number of windows in every frame.
- A frame is split into windows. A frame is moved along the data, a window at a time. Normally acknowledgements are only sent back when a window is full and not for every chunk.
- A window is comprised of the number of chunks that are sent from source to destination. Each chunk can contain data from one or more files.
- A chunk defines the maximum MQ message size.

If the network bandwidth is adequate, you can increase the agentChunkSize value that is actually file message size, for example, agentChunkSize is raised from a default value of 256 KB to 1 MB in LAN, as shown in Table 13-3 on page 343.

Table 13-4 Transfer limit properties

Property name	Default value	Description
maxFilesForTransfer	5000	The maximum number of files that are allowed for a transfer. If a transfer specifies more files than the value of maxFilesForTransfer, the additional file transfers fail. 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.
maxSourceTransfers	25	The maximum number of concurrent transfers that are allowed for a source agent. Attempts to start additional transfers fail if the source agent has reached the limit that is specified by the maxSourceTransfers property.

Property name	Default value	Description
maxDestinationTransfers	25	The maximum number of concurrent transfers that are allowed for a destination agent. Attempts to start additional transfers fail if the destination agent has reached the limit that is specified by the maxDestinationTransfers property.

If the system resources of CPU, I/O, and memory are available, you can raise the maxSourceTransfers and maxDestinationTransfers value for more concurrent threads, as shown in Table 13-4 on page 344.

13.3.3 Operating system parameters

Because WebSphere MQ File Transfer Edition is developed in Java, you must pay attention to use agents to transfer large files on UNIX and Linux. If you use the default user resource parameters on UNIX and Linux, some exceptions or core dumps might occur when large files are transferring. We recommend that you increase the values of some FTE user resources to *unlimited* on UNIX and Linux environments.

Example 13-14 shows the improved parameters on AIX. You can separately set your user parameters of size, cpu, data, rss, and stack to the value of -1 (which means unlimited).

Example 13-14 User resource parameters on AIX

```
AIX53:/>more /etc/security/limits
*
* Sizes are in multiples of 512 byte blocks, CPU time is in seconds

* fsize            --- soft file size in blocks
* core             --- soft core file size in blocks
* cpu              --- soft per process CPU time limit in seconds
* data             --- soft data segment size in blocks
* stack            --- soft stack segment size in blocks
* rss             --- soft real memory usage in blocks
* nofiles          --- soft file descriptor limit
* fsize_hard      --- hard file size in blocks
* core_hard       --- hard core file size in blocks
* cpu_hard        --- hard per process CPU time limit in seconds
* data_hard       --- hard data segment size in blocks
* stack_hard      --- hard stack segment size in blocks
* rss_hard       --- hard real memory usage in blocks
* nofiles_hard    --- hard file descriptor limit
```

```
default:
        fsize = 2097151
        core = 2097151
        cpu = -1
        data = 262144
        rss = 65536
        stack = 65536
        nofiles = 2000
wmbuser:
        fsize = -1
        core = 2097151
        cpu = -1
        data = -1
        rss = -1
        stack = -1
        nofiles = 3000
```

You use the **`ulimit`** command to modify user resource parameters on Linux, as shown in Example 13-15.

Example 13-15 User resource parameters on Linux

```
[wmbadmin@newyork security]# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 8192
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 8192
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

13.3.4 Shared disks

When the files are read from file systems or disks on source agents and written to file systems on destination agents, the I/O is very busy. So we suggest that you transfer files based on multiple high-speed shared disks or storage. Locate the WebSphere MQ log and queue manager data in the shared disks or storage. Because of file transfer recovery, FTE uses some persistent queues and status messages. I/O will be busy when the queue buffer size is not enough and at the same time there is a bulk of files above 1 GB size.

14

Integration

In this chapter, we cover several of the strategies that are available for using WebSphere MQ File Transfer Edition (FTE) to expand the integration and management of your file transfer systems. This integration can be accomplished both by expanding the basic MQ infrastructure and through effectively using the integration interfaces that are available to FTE.

The topics that we discuss in this chapter are:

14.1 Integration overview

Currently, many business critical applications connect by exchanging files. Most organizations have several products and different techniques for performing file transfers. Typically, there is a mix of FTP, locally developed programs, and other file transfer products, as illustrated in Figure 14-1.

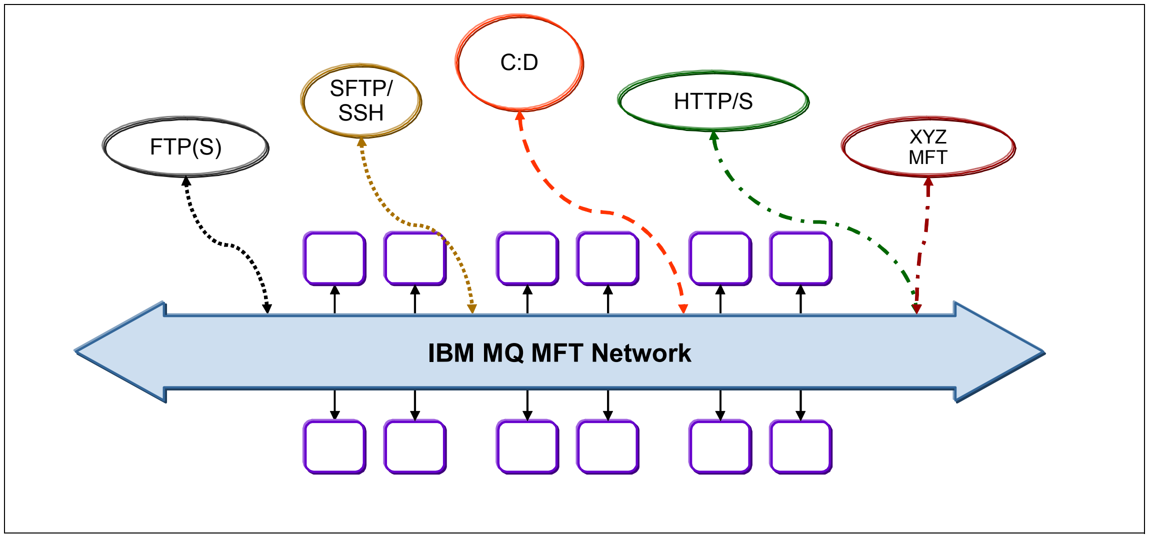


Figure 14-1 Multiple file transfer products

If you want to take advantage of the WebSphere MQ FTE managed file transfer (MFT) integration over an MQ network, in this chapter, we discuss approaches that you can use to avoid a complete rip-and-replace of your existing solutions. There are techniques available that allow an incremental introduction of FTE while at the same time interoperating with existing solutions.

14.2 Expanding your WebSphere MQ network

This integration approach is so obvious that it might easily be overlooked: simply expand your MQ network. In addition to supporting managed file transfer, there are many advantages to simply expanding the reach of your existing WebSphere MQ network to allow WebSphere MQ File Transfer Edition to be more broadly used.

WebSphere MQ:

- Supports the broadest range of APIs, programming languages, and operating system platforms
- Provides the only JMS provider that can be implemented on any standards-compliant JEE server
- Provides rich Web services interfaces that meet customer needs for WS-Reliability
- Offers a broad range of qualities of service and messaging methods, including publish/subscribe
- Provides Low Latency and Extended Security editions
- Offers the most scalable, most manageable messaging system available
- Assures transactional message delivery end-to-end

Therefore, as your first integration option, consider simply expanding the reach of your MQ network.

14.2.1 MQ FTE server agents

You can install an FTE server agent on any system that has an MQ queue manager. Expanding your MQ network, will, in addition to many other benefits, also allow you to install additional FTE server agents.

Investigate those platforms that already have MQ server Version 6.0.2.4 or better installed—or upgrade. FTE server agents are currently supported on the following platforms:

- AIX
- HP-UX
- Linux for x86
- Linux for System z
- Solaris
- Windows, and
- z/OS

For detailed platform support information, go to:

http://www-01.ibm.com/software/integration/wmq/filetransfer/requirements/index.html#v711

The next step to consider in implementing managed file transfer integration is to install additional MQ FTE server agents.

14.2.2 MQ FTE client agents

After you establish an MQ network of MQ servers, you might also add FTE clients to those machines that do not have MQ installed. These machines can then participate in managed file transfer by connecting to an MQ server that is located elsewhere in the network.

An FTE client agent can be installed on any supported platform and does not require that MQ be present. Using an FTE client, you can take advantage of an FTE agent's ability to participate in managed file transfer with only a small installation footprint.

After you install additional FTE server agents, consider installing, where possible, FTE client agents.

14.3 Integration beyond FTE

After you install FTE agents where you can, you might still need to consider the following questions:

- How can you migrate to FTE without a complete *rip and replace*?
- How do you deal with platforms not supported by FTE?
- How can you exchange files with external partners?
- How you can connect using other file transfer protocols?
- How can your managed file transfer (MFT) network reduce your e-mail attachment load and security risk?

In the following sections, we discuss some approaches to solving these problems.

14.4 Common application integration patterns

There are two common patterns of integration of FTE with processing applications:

- Source side: Run a process prior to the file transfer. Typically, this either generates or manipulates the file that is being transferred
- Destination side: Run a process after the file transfer. Typically, this consumes or post-processes the transferred file.

14.4.1 Source-side integration

There are three basic options for source-side application integration:

- Have the application write files into a directory monitored by a sending agent. Trigger files are commonly used to indicate that the file to-be-transferred completely arrived.
- Have the application send an XML command message to a sending agent thereby initiating a transfer.
- Have a sending agent run the data generation of the manipulation application prior to starting the transfer.

14.4.2 Destination-side integration

There are three basic options for destination-side application integration:

1. Have the sending agent transfer the file into a receiving directory that is monitored by the application.Trigger files are commonly used to indicate that the received file completely arrived.
2. Have the receiving agent run an application that processes the transferred file. This approach is used for programs that are quick to start and that run for a short period of time.
3. Have the receiving agent run a program that notifies the application to process the transferred file. This approach is used if the called application takes a long time to start or a long time to execute. In that case, it might be better to have the application running continually in the background and have the agent notify if there is a file that is available to process.

14.5 Using designated directories

A straightforward approach to integrating your FTE agents with foreign protocols, and other file transfer products, is to use designated directories for handing off files between FTE and other products and protocols, as shown in Figure 14-2 on page 354.

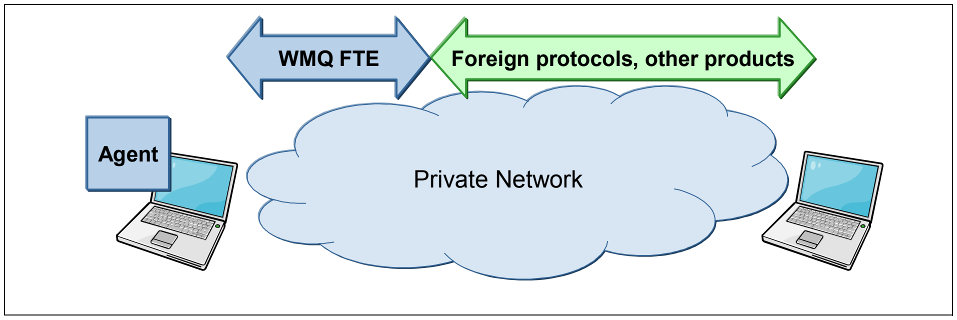


Figure 14-2 The problem of integrating FTE agent with foreign protocols

Designated directories are located at the junction of the two arrows in Figure 14-2. From an FTE perspective, there should be one or more locations where file data can be exchanged. These locations have an FTE agent.

From a non-FTE perspective, there is also one or more locations where file data can be exchanged. These locations run the software that is appropriate for the non-FTE product.

There is a good example of this approach in Chapter 9, "Phase 3: Complex transfers" on page 227, where FTE hands off a file to WebSphere Message Broker and Message Broker hands off a file to FTE—all as part of a completely automated MFT solution.

14.6 Scripting with fteAnt

Apache Ant is a powerful scripting language that you can use to automate and extend the reach of FTE. As of FTE release v7.0.1, Ant is automatically installed as part of a regular installation. You can access Ant to use with FTE by issuing the **`fteAnt`** command.

14.6.1 Creating a bridge to other managed file transfer products

You can configure the **`fteAnt`** command to create a bridge to other file transfer products. Use the **`fteAnt`** functions to automatically move files between an internal FTE network and an external organization that is using another managed file transfer product, as shown in Figure 14-3 on page 355.

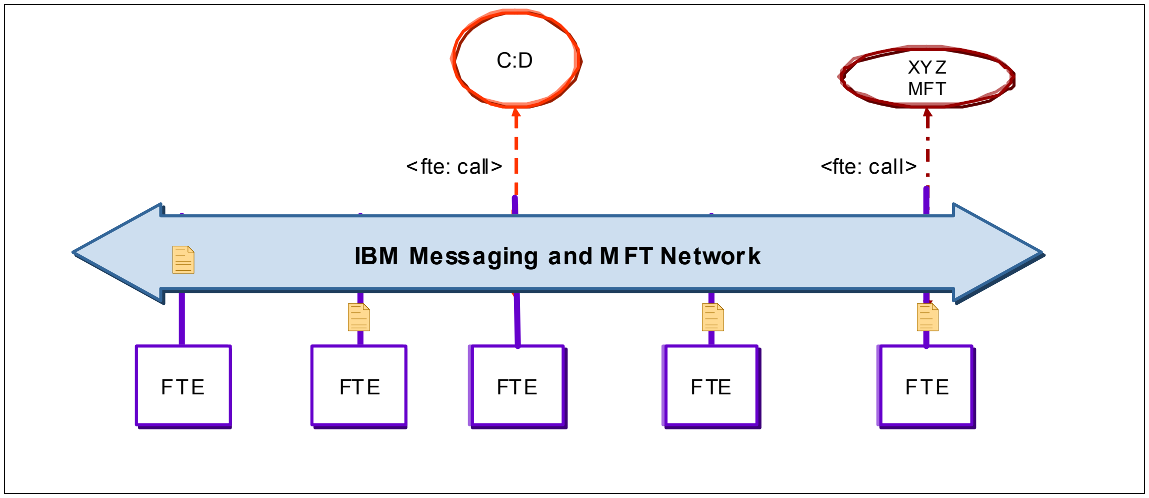


Figure 14-3 Creating an fteAnt bridge to other managed file transfer products

All that is required for this to work is to set up a platform on which both an FTE node and another product's node are running. You can then configure **`fteAnt`** to automatically bridge between the two.

Logs of the FTE portion of each of the above transfers are published to the SYSTEM.FTE topic on the coordination queue manager. If the FTE database logger is installed, these logs can be persisted in a database.

In addition, the **`fteAnt`** bridge scripts can invoke any required exception handling, such as e-mail notifications, should something not work as expected.

This integration solution is transparent to users of both FTE and the other managed file transfer product.

All of this is possible because, as a basic capability, **`fteAnt`** can, without programming, call any executable. Therefore, it can call any managed file transfer product for which there is a command line interface.

Note: For further information about using the **`fteAnt`** call task, see the Call topic in the FTE Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/call.htm`

14.6.2 Calling other file transfer protocols using cURL

Another example of how you can use **`fteAnt`** to accomplish integration is to call a product, such as cURL. The cURL product can create a bridge between FTE and a wide range of file transfer protocols.

What is cURL?

cURL is a widely used Open Source/Free Software command line tool for transferring files with URL syntax. cURL is useful for file transfer scripting because a single cURL command can push or pull files and other resources to or from remote systems.

Platforms supported

cURL supports the following platforms:

- Any Linux distribution
- Win32®, Win64
- AIX
- HPUX
- z/OS
- Solaris
- VMS
- SCO Unix
- SGI IRIX
- Tru64 UNIX
- Mac

File transfer protocols supported

cURL supports the following transfer protocols:

- FTP, FTPS, SFTP, TFTP
- HTTP, HTTPS
- SCP, TELNET, DICT

cURL provides excellent support for protocols, including check point restart support if the server supports this feature.

Security and authentication methods

cURL provides excellent support for the following forms of security and authentication:

- HTTP Basic Authentication (user+password)
- HTTPS (SSL certificates)
- Proxy Tunneling
- Cookies
- Basic FTP type (user+password) authentication with netrc-file support.

- Digest
- NTLM
- Negotiate
- Kerberos

cURL extends the reach of the FTE managed file transfer

Using **fteAnt**, you can write scripts that include calls to cURL and doing this makes a broad range of managed file transfer solutions possible. You can bridge to other file transports. Using cURL, FTE can push or pull files to and from FTP type solutions, including third party products, as shown in Figure 14-4.

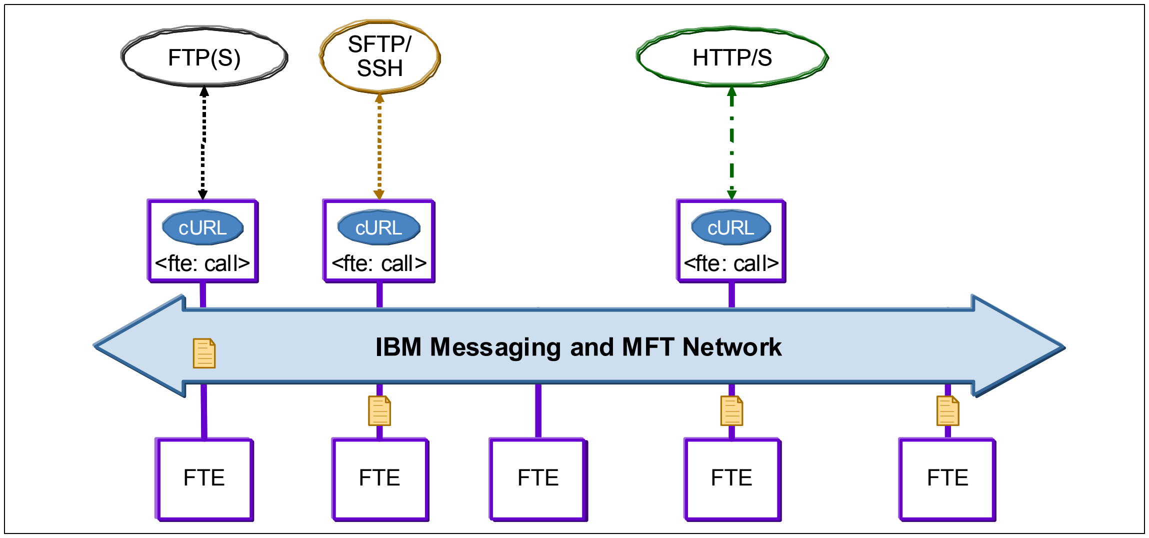


Figure 14-4 Using cURL to extend the reach of FTE

Implementing cURL with fteAnt

Using the **fteAnt** "call" function, you can use an XML command file that marshals the required parameters to cURL arguments. Your FTE agent then does a controlled launch of any cURL command.

The **fteAnt** call function can be set up with parameters to retry a failed or incomplete transfer.

The benefits of using cURL with fteAnt

The benefits of using cURL in an fteAnt script are:

- Enables a non-programmatic solution to extend the reach and range of FTE beyond the MQ boundaries.

- Extends the transfer and automation capabilities of FTE to other file transfer transports.
- Provides a *node at a time* migration approach. FTE can be implemented incrementally.
- Provides a bridging solution to third-party protocols.
- Allows the use of FTE as an enterprise solution regardless of file transfer protocol.

Examples

Chapter 9, "Phase 3: Complex transfers" on page 227 of this book contains several examples of using **`fteAnt`** with cURL to solve integration problems.

14.6.3 Overview of the fteAnt integration capability

Figure 14-5 illustrates the fteAnt integration capabilities that we discussed.

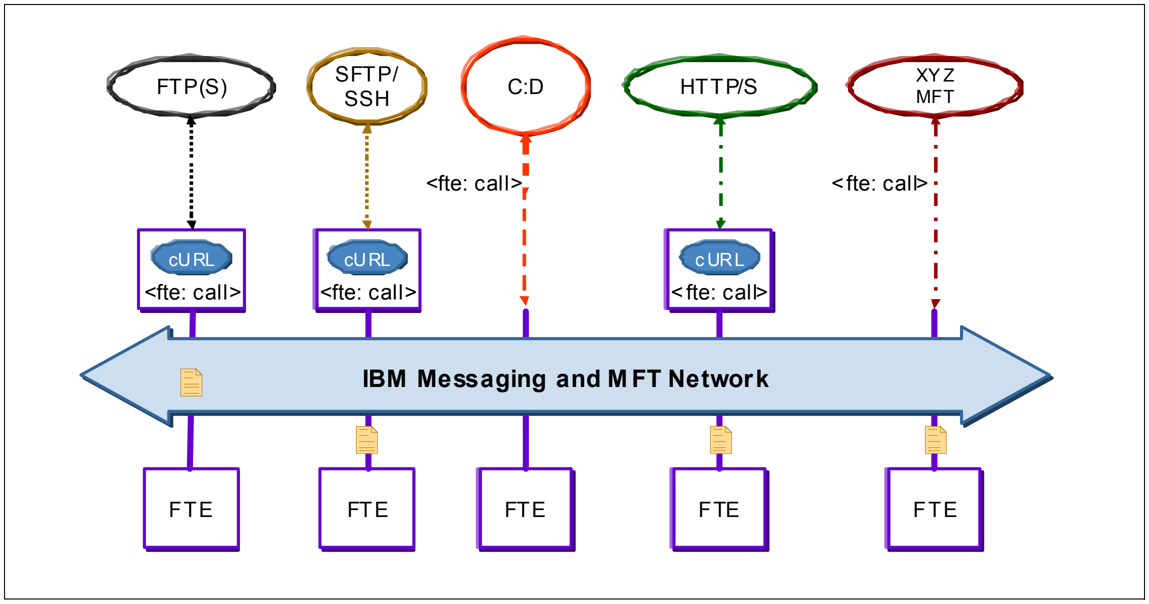


Figure 14-5 fteAnt integration capabilities

Note: For further information, see "Using Apache Ant with WebSphere MQ File Transfer Edition" in the FTE Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/using_ant.htm`

14.7 FTE resource monitor's <managedCall>

Using the resource monitoring feature of WebSphere MQ File Transfer Edition, you can start and coordinate tasks, such as file transfers, based on some resource condition that is being satisfied.

The resource monitoring feature of WebSphere MQ File Transfer Edition allows you to start and coordinate tasks, such as file transfers, based on some resource condition being satisfied.

A common scenario is to monitor a directory for the presence of a ready file or trigger file. An external application might be processing multiple files and placing them in a known source directory, such as the designated directory that we discussed in 14.5, "Using designated directories" on page 353. When the application completes its processing, it indicates that the files are ready to be transferred, or otherwise acted upon, by placing a ready file into the monitored location.

Of importance to integration is the fact that you can use the monitor's <managedCall> element to configure the monitor to automatically call other commands from the monitor agent, including executable programs, such as cURL or other managed file transfer applications, Ant scripts, and JCL jobs. To call commands, you can edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties. We provide several examples of using <managedCall> in Chapter 9, "Phase 3: Complex transfers" on page 227.

Using the monitor's <managedCall> element you can automate the two integration techniques that we discussed in this chapter: 14.5, "Using designated directories" on page 353, and 14.6, "Scripting with fteAnt" on page 354.

Note: For further information, see "Configuring monitor tasks to invoke commands and scripts" in the FTE Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/configuring_monitor_tasks.htm

14.8 Managed file transfer for SOA using an ESB

Thus far, we showed two basic methods for initiating an FTE file transfer:

- MQ Explorer GUI interface
- Command line

FTE capabilities, such as simple transfers, scheduled transfers, fteAnt scripts, and resource monitors, have all been implemented using these methods.

There is a third method that opens up a wide range of enterprise integration options: you can put an FTE XML file transfer command message directly on an FTE agent's command queue. (Although not explicitly discussed, this method was demonstrated in 9.2.5, "Running the package distribution command automatically using an FTE resource monitor" on page 235 in our tests using RFHUTILC.)[1]

The ability to initiate an FTE file transfer by placing a properly formatted XML message on a queue, allows any authorized application, including an application that is exposed as a service, to initiate a file transfer programmatically.

It is important to mention that placing a command message directly on an agent's command queue provides the same FTE managed file transfer (MFT) capabilities as when using either the MQ Explorer GUI or the command line:

- FTE agents respond to commands by publishing status and log messages on the SYSTEM.FTE topic.
- Users, administrators, and applications can subscribe to and utilize this information.

In this section, we briefly discuss how a managed file transfer service, initiated by placing an XML message on a queue, can be flexibly deployed in a service-oriented architecture (SOA) using an enterprise service bus (ESB).

14.8.1 SOA with an ESB

At the most basic level, an SOA with an ESB consists of the following components, as shown in Figure 14-6 on page 361:

- Service provider
- Service consumer
- Service registry

[1] See, "File transfer request message format", in the FTE Information center at: http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/transfer_message_format.htm

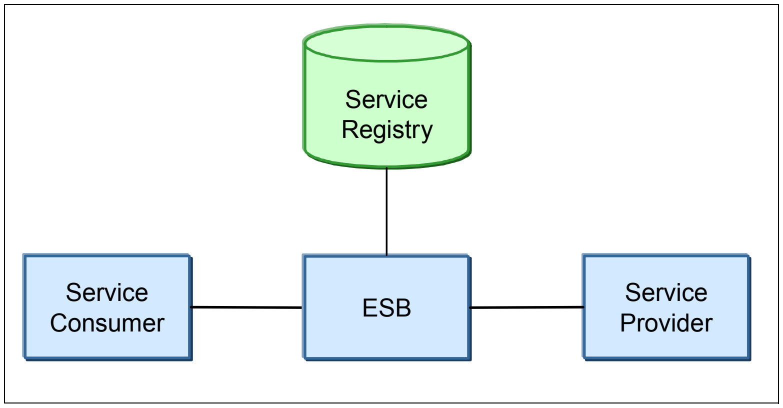


Figure 14-6 Basic SOA architecture with an ESB

In this architecture, the ESB provides the service consumer with a standards-based interface to back end services. Additionally, the service registry provides for service governance, service availability information, and service-related metadata.

14.8.2 Managed file transfer example using an ESB

Figure 14-7 shows an example implementation of an SOA as applied to managed file transfer with FTE.

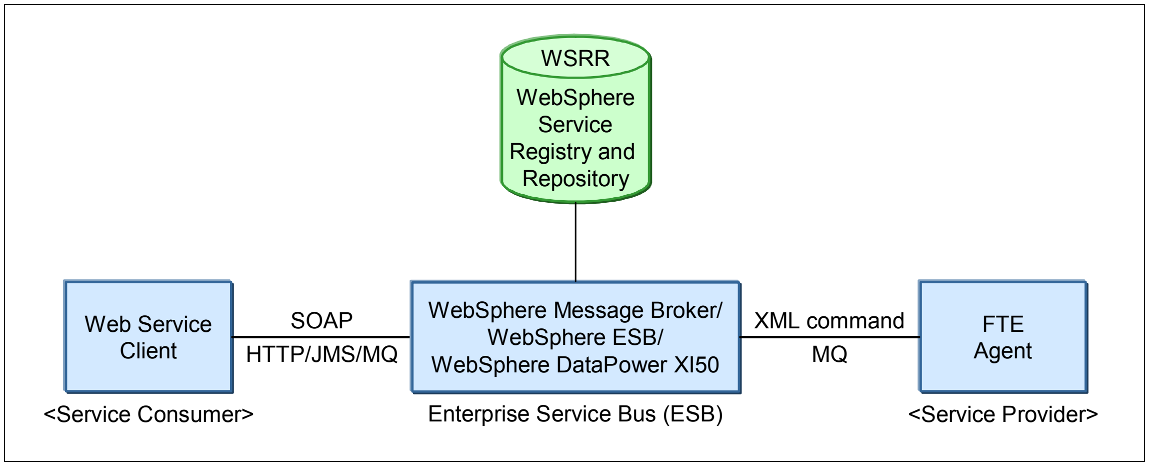

WSRR
WebSphere Service Registry and Repository
Web Service Client
SOAP
HTTP/JMS/MQ
WebSphere Message Broker/ WebSphere ESB/ WebSphere DataPower XI50
XML command
MQ
FTE Agent
<Service Consumer>
Enterprise Service Bus (ESB)
<Service Provider>

Figure 14-7 Managed file transfer example

In Figure 14-7 on page 361:

1. The service consumer, a Web service client, initiates a file transfer request by sending a SOAP message over either HTTP(s), JMS, or MQ.
2. The message is received by one of three alternative ESBs: WebSphere Message Broker, WebSphere ESB, or the WebSphere DataPower® XI50 appliance.[2]
3. The WebSphere Service Registry and Repository provides the ESB with real-time information regarding existing file transfer services, including information regarding available file transfer agents and the underlying MQ transport infrastructure.[3] [4]
4. The ESB matches the current SOAP request with an appropriate available file transfer service. The implementation details, for example, source and destination agent names, are completed based upon information provided by the registry.
5. The ESB then generates an FTE XML command message and sends it to the designated FTE source agent for execution.[5]

Note: A detailed example of how to set up this architecture using the DataPower XI50 appliance as the ESB is in the developerWorks article *Managed file transfer for SOA: A complete solution using WebSphere DataPower, WebSphere MQ, and WebSphere Service Registry and Repository*, which is located at:

http://www.ibm.com/developerworks/websphere/library/techarticles/0907_amato/0907_amato.html

[2] A discussion of how to select an appropriate ESB is contained in the developerWorks® article, “Enterprise Service Bus implementation patterns” at the following URL:
http://www.ibm.com/developerworks/websphere/library/techarticles/0712_grund/0712_grund.html

[3] An introduction to the WebSphere Service Registry and Repository may be found in the developerWorks article, “Introducing the IBM WebSphere Service Registry and Repository, Part 1” at the following URL:
http://www.ibm.com/developerWorks/websphere/library/techarticles/0609_mckee/0609_mckee.html

[4] SupportPac FA01 is a service discovery plug-in for FTE and WSRR. It polls the coordination queue manager for information about the agents that exist in an FTE network. Using the data retrieved from the agents’ retained publications, it is able to build up a map between queue managers and agents. FA01 may be found at the following URL:
http://www.ibm.com/support/docview.wss?rs=171&uid=swg24022956&loc=en_US&cs=utf-8&lang=en

[5] See, “File transfer request message format”, in the FTE Information center at:
http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.admin.doc/transfer_message_format.htm

14.8.3 Benefits of managed file transfer for SOA using an ESB

The wide range of FTE tools that are provided are primarily intended for direct interaction with the user. They require either the FTE GUI or command-line API.

FTE's ability to directly accept XML command messages, when combined with the use of an SOA, provides a powerful, flexible, platform-independent, and standards-based enterprise-wide managed file transfer implementation.

The benefits are:

- Business benefits:
  - Flexibility and agility
  - Improved time-to-value/time-to-market
- Technical benefits:
  - Less initial coding
  - Cheaper to maintain
  - Interoperability
  - Automatic program-to-program invocation
- Architectural benefits:
  - Metadata-based (WSRR)
  - Standards-based (Web services)
  - Loosely-coupled (WSRR, ESB, Web services)

14.9 Exchanging files with external partners

Exchanging files with external partners requires the use of business-to-business (B2B) standards and strong *trading partner* profile management. Figure 14-8 shows an example of such a scenario.

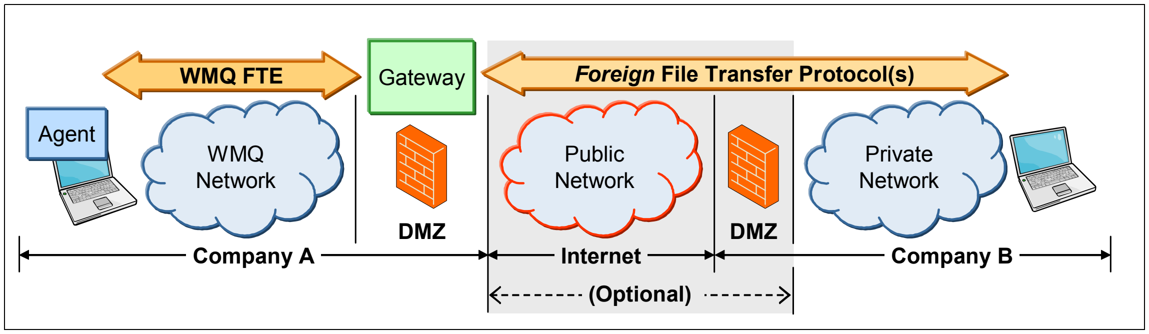


Figure 14-8 Exchanging files with external partners

In the scenario illustrated in Figure 14-8 on page 363:

- Company A and B periodically exchange file data. This exchange needs to take place over zero or more intermediate networks, for example, the Internet.
- Internally, company A likes the idea of using FTE to move all file data. Company B, however, uses a foreign file transfer solution to move file data from Company A's gateway into its own private network.
- Inside Company A, there is at least one location where file data can be exchanged. These machines run an FTE agent.
- Inside Company B, there is at least one location where file data can be exchanged. These locations are not running an FTE agent. They are running software that supports the foreign file transfer solution that Company B is using.

Management of the data exchange between Company A's DMZ and Company B has to be based on B2B standards and requires strong Trading Partner profile management. This is therefore *not* a good fit for **`fteAnt`** with cURL.

This situation typically requires a B2B product, for example WebSphere Partner Gateway, WebSphere Transformation Extender, or WebSphere DataPower. Using such a B2B product, Company A and B's data exchanges are then mediated by the B2B solution, for example, FTE and the foreign file transfer software each exchange files with WebSphere Partner Gateway.

14.10 Summary

There are several effective ways that we discussed that allow FTE to expand the integration and management of your file transfer systems. Among those are:

- Expanding your MQ network
- Using FTE clients
- Using designated directories for file hand-off
- Using the **`fteAnt`** powerful scripting and execution management capabilities, for example, creating a bridge to other managed file transfer products or calling other protocols using cURL
- Using the FTE resource monitor's <managedCall> element to automate the use of designated directories and calls to executable programs, such as cURL and other managed file transfer applications, Ant scripts, and JCL jobs.

- Implementing managed file transfer for SOA using an ESB allows for a flexible enterprise-wide, platform-independent, standards-based implementation
- Using products, such as WebSphere Partner Gateway, to mediate exchanges between organizations

New capabilities can be added to FTE's integration capabilities. Check the FTE product Web site to stay informed of the latest available integration features:

`http://www.ibm.com/software/integration/wmq/filetransfer/`

Part 4

Appendixes

A

Additional material

This section refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG247760`

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247760.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Abbreviations and acronyms

CA	Certification Authority
HA	High availability
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
JKHL	JKHL Travel
MCA	Message Channel Agent
MFT	Managed File Transfer
MQMD	Message Descriptor Structure
MQOD	MQ Object Description
RRS	Resource Recovery Service
SAF	Security Access Facility
SFTP	SSH File Transfer Protocol
SSL	Secure Socket Layer
zAAP	z Application Assist Processor

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 373. Note that some of the documents referenced here might be available in softcopy only:

- *WebSphere MQ V6, WebSphere Message Broker V6, and SSL*, REDP-4140

Online resources

These Web sites are also relevant as further information sources:

- Apache Ant

  `http://ant.apache.org/`
- cURL

  `http://curl.haxx.se/`

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Getting Started with WebSphere MQ File Transfer Edition V7

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Getting Started with WebSphere MQ File Transfer Edition V7

Discover how to install, configure, and administer managed file transfer solutions

Identify how to automate transfers and integrate your enterprise

Learn by example with practical scenarios

WebSphere MQ File Transfer Edition provides an enterprise-ready managed file transfer capability that is both robust and easy-to-use. WebSphere MQ File Transfer Edition exploits the proven reliability and connectivity of WebSphere MQ to transfer files across a wide range of platforms and networks.

In this IBM® Redbooks® publication, we provide a complete guide for getting started with WebSphere MQ File Transfer Edition. In Part one of the book, we provide a technical overview of the product and provide installation, configuration, and administration guidance for distributed and z/OS® platforms.

In Part two, we provide a series of scenarios to show how you can use WebSphere MQ File Transfer Edition to create managed file transfer solutions. These scenarios range from simple point-to-point transfers through to resource monitoring and complex scripted transfers that include the use of Apache Ant. We describe each scenario step-by-step, which allows you to follow along in your own environment.

In Part three of this book, we discuss topics, such as security, user exits, and the integration of WebSphere MQ File Transfer Edition with other products and solutions.

SG24-7760-00 ISBN 073843325X