



StarTeam 16.3

ActiveMQ MPX Administrator's
Guide

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
<http://www.microfocus.com>

Copyright © Micro Focus 2018. All rights reserved.

MICRO FOCUS, the Micro Focus logo and StarTeam are trademarks or registered trademarks of Micro Focus IP Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.

All other marks are the property of their respective owners.

2018-03-27

Contents

Preface	5
Documentation	5
Contacting Support	6
Overview	7
Framework and Architecture	8
MPX Components	10
Component Descriptions	13
ActiveMQ MPX Security	14
Data Encryption	14
User Authentication and Access Rights	14
Installation	15
Component Configuration	15
Dependencies - Startup Order for MPX Components	17
Configuring MPX to use ActiveMQ MPX	18
Requirements When Using StarTeamMPX and ActiveMQ MPX Together	19
Requirements When Using ActiveMQ MPX Only	20
Managing Message Brokers	21
Planning for Message Brokers	21
Understanding Clouds	21
Message Broker Communication	22
Message Routing in Message Broker Clouds	22
Routing in Unconnected Message Broker Clouds	23
Volume Considerations	24
Using Message Brokers with a Firewall	25
Configuring a Message Broker	25
Configuring a Message Broker Cloud	26
Changing the Endpoint of a Message Broker	26
Configuring Two Message Brokers in a Fail-Over Configuration	27
Enabling Tracing for Message Brokers	28
Controlling Connections	28
Managing the Transmitters	29
Configuration-specific Transmitter XML Files	29
Enabling Transmitters for Server Configurations	29
Enabling MPX on Multiple StarTeam Server Configurations	30
Understanding Connection Profiles	31
Understanding the Event Transmitter	31
Event Transmitter Startup	31
Event Transmitter XML File Format	32
Using Profiles with Multiple Connections	35
Understanding the File Transmitter	36
File Transmitter Startup	37
File Transmitter XML File Format	37
Managing MPX Cache Agents	38
Planning for the MPX Cache Agents	38
MPX Cache Agent Operations	39
Configuring a Root MPX Cache Agent	40
Configuring a Remote MPX Cache Agent	41
Cache Agent XML Parameters	42
Parameters Used by Any MPX Cache Agent	42
Parameters Used by Remote MPX Cache Agent	46

Parameters Used by Root MPX Cache Agent	48
Reviewing Status and Log Information	49
Using MPX Cache Agent with the Clients	49
Object Caching	50
How Object Caching Works	50
Components Needed for Object Caching	52
Configuring Object Caching	52
Configuring Clients	58
Using ActiveMQ MPX from a Client	58
Displaying MPX Status	58
Choosing a Non-default Connection Profile	59
Logging MPX Information in the Client Log	59
Using MPX Cache Agent from the StarTeam Cross-Platform Client and IDEs	60
Enabling MPX Cache Agent Use	60
Checking out Files with the MPX Cache Agent	61
Using MPX Cache Agent with Bulk Checkout Utility	62
Running MPX Components	63
Running Message Broker on Microsoft Windows	63
Starting a Message Broker	63
Stopping a Message Broker	63
Running the ActiveMQ MPX Message Broker on Linux	64
Running MPX Cache Agents	64
Running MPX Cache Agent On Microsoft Windows	64
Running MPX Cache Agent on Linux	66
Server Log Entries	67
Start-Up Messages	67
Reconnect Messages	67
Troubleshooting ActiveMQ MPX	69
Diagnosing a Message Broker	69

Preface

This manual contains information for StarTeam administrators who manage MPX-enabled server configurations. It explains the basic operation and architecture of a ActiveMQ MPX system, and provides instructions for installing and configuring the ActiveMQ MPX components. For information about performing other administrative tasks on a StarTeam Server configuration, see the *StarTeam Server Help*. For information about installing ActiveMQ MPX components and system requirements, refer to the *StarTeam Installation Guide*.

Documentation

The documentation is your guide to using the product suite. StarTeam documentation is provided in several formats: online help, HTML, and Adobe PDF. Documentation is available from the **Help** menu within the product.

If you are using a Microsoft Windows system, you can locate documentation for the products by clicking **Start > Programs > StarTeam > <Product> > Documentation**. The Documentation menu lists all of the available documentation for the selected product.

You can also download documentation directly from: <http://supportline.microfocus.com/productdoc.aspx>.

HTML Documentation

Readme files can be found directly under the root installation directory (or on the root of the installation CD). For documentation available in other languages (Japanese, French, or German), the language-specific versions of the release notes are indicated with an appropriate `_countrycode` in the filename. For example, `readme_ja.html` contains release note information for the Japanese language. PDF manuals are located in the `Documentation` subfolder on the product CDs.

Adobe PDF Manuals

The following documentation is provided in Adobe PDF format. All manuals distributed in Adobe Acrobat (.PDF) format require Adobe Acrobat Reader to display them. The installation program for Adobe Acrobat Reader is available from the Adobe web site at: www.adobe.com.

Release Notes	Contains system requirements and supported platforms for the products.
StarTeam Installation Guide	The StarTeam Installation Guide contains detailed instructions for installing and configuring the core StarTeam products.
StarTeam Server Help	This manual is identical to the online help version.
StarTeam Cross-Platform Client Help	This manual is identical to the online help version.
StarTeam Command Line Tools Help	Explains how to use the command-line tools and provides a reference for the various commands.
StarTeam File Compare/Merge Help	This manual is identical to the online help version of the StarTeam File Compare/Merge help.
StarTeam Workflow Extensions User's Guide	Explains how to design and manage StarTeam Extensions such as alternate property editors (APEs). It also covers the StarTeam Workflow Designer and StarTeam Notification Agent.

StarTeamMPX Administrator's Guide

Explains the basic operation and architecture of the system, and presents instructions on installing and configuring the components.

ActiveMQ MPX Administrator's Guide

Explains the basic operation and architecture of the system, and presents instructions on installing and configuring the components.



Note: Depending upon which products you purchased and installed, not all of the application manuals will be on your system.

Contacting Support

Micro Focus is committed to providing world-class services in the areas of consulting and technical support. Qualified technical support engineers are prepared to handle your support needs on a case-by-case basis or in an ongoing partnership. Micro Focus provides worldwide support, delivering timely, reliable service to ensure every customer's business success.

For more information about support services, visit the Micro Focus SupportLine web site at <http://supportline.microfocus.com> where registered users can find product upgrades as well as previous versions of a product. Additionally, users can find the Knowledge Base, Product Documentation, Community Forums, and support resources.

When contacting support, be prepared to provide complete information about your environment, the product version, and a detailed description of the problem, including steps to reproduce the problem.

For support on third-party tools or documentation, contact the vendor of the tool.

Overview

ActiveMQ MPX is a framework for publish/subscribe messaging. It contains both common and application-specific components that together provide advanced messaging capabilities.

StarTeam Enterprise licenses support the following MPX components:

- MPX Message Broker
- MPX Event Transmitter

StarTeam Enterprise Advantage licenses support all of the MPX components:

- MPX Message Broker
- MPX Event Transmitter
- MPX File Transmitter
- MPX Cache Agent

ActiveMQ MPX improves the performance of the clients and extends the scalability of server configurations. When the term client is used in this guide, it refers to any client that can take advantage of one or more ActiveMQ MPX features.

Changes to the server configuration's repository are broadcast in an encrypted format to StarTeam clients and MPX Cache Agents through a publish/subscribe channel. The MPX Event Transmitter broadcasts encrypted messages about changes to objects, such as change requests, and the MPX File Transmitter broadcasts archive files.

Caching modules automatically capture events that a client subscribes to. This reduces the client's need to send refresh requests to the server and improves client response times for the user.

You can install and configure MPX Cache Agents to cache files and/or objects in a network-near location to speed up check-out operations. They reduce the distance that the data travels at the time of the client check-out operation. While MPX Cache Agents are MPX clients that rely on messages and files transmitted by the File Transmitter, they also serve other MPX clients as they check out files.

The ActiveMQ MPX technology offers a number of key benefits, including:

Bandwidth multiplication	<p>Every request by a client that is fulfilled from a cache is a request that the server does not have to fulfill. As a result, a single server configuration can support more clients.</p> <p>In certain environments, requests for item refresh may constitute up to 50% of client-to-server traffic. Refresh requests increase with project size and when the "All Descendants" option is used.</p> <p>For the person who builds software products, check-out operations are a very high percentage of the client-to-server traffic. Using the <code>Bulk CheckOut (bco)</code> command-line utility with a MPX Cache Agent can speed up the time it takes to create a build.</p>
Performance acceleration	<p>Because the event caching modules reside on the same computer as the client and the MPX Cache Agents are on computers that are network-near the supported clients, requests filled from the caches are faster than those requiring a round-trip to the server.</p>
Burst control	<p>As the number of clients increases, the likelihood of burst periods increases, during which time a server configuration can become deluged and less responsive. The caching modules even-out the demand on a server configuration.</p>

The following lists the MPX features of which specific clients can take advantage.

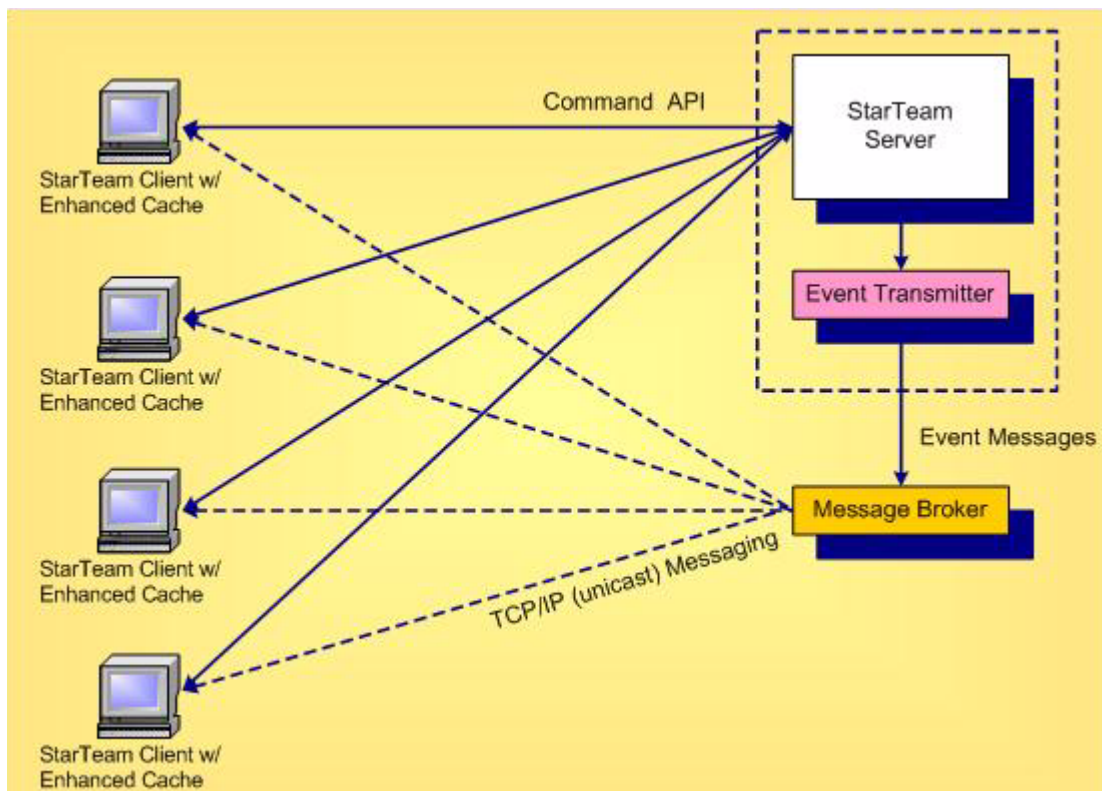
- StarTeam Cross-Platform Client - Event and file/object caching.
- Bulk CheckOut (bco) command-line utility - File caching.
- IDEs based on StarTeam Cross-Platform Client and .NET components (such as the Eclipse integration and the Visual Studio .NET integration) - Event and file/object caching.

Framework and Architecture

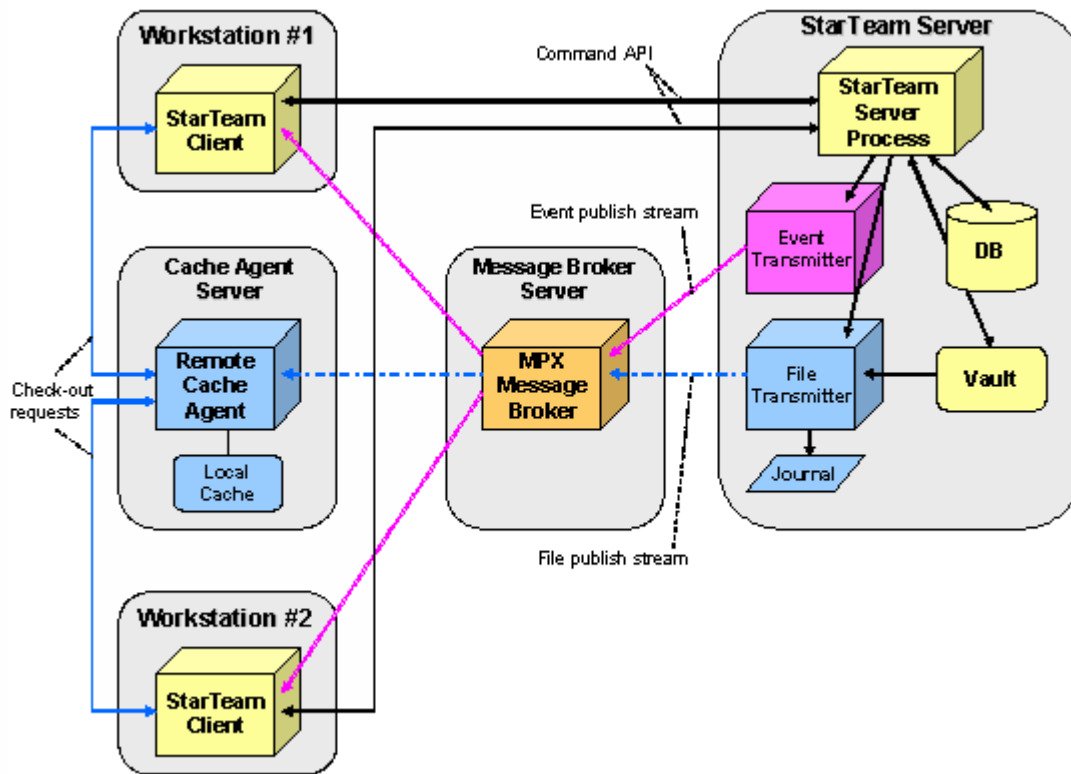
The key components within the ActiveMQ MPX framework are messaging engine components, publisher components, and subscriber components. Some components both subscribe and publish.

Messaging engine component	The MPX Message Broker is the primary messaging engine, providing “unicast” messaging.
Publisher components	The publisher components send messages to messaging engines, which forward those messages to the appropriate subscriber components. For example, the MPX Event Transmitter and MPX File Transmitter are publishing components.
Subscriber components	The subscriber components receive only those messages to which they have subscribed. Subscriber components receive messages through TCP/IP from the MPX Message Broker. StarTeam clients can receive and cache event messages.

The following presents an overview of the ActiveMQ MPX system architecture for event messaging. Depending upon the individual needs of a particular site or facility, there may be several MPX-enabled server configurations and MPX Message Brokers serving many clients.



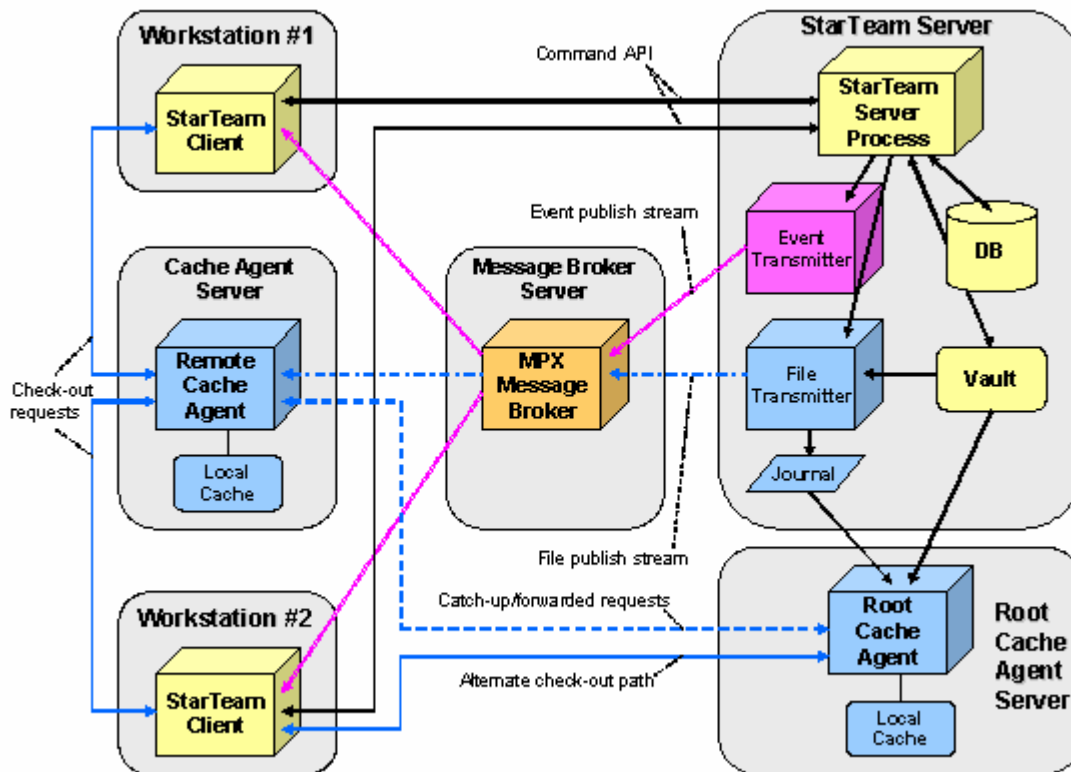
The following present an overview of the ActiveMQ MPX system architecture for file transmission. In the first, a single MPX Cache Agent is operating on its own server, servicing check-out requests for two MPX Cache Agent-aware clients. Depending upon the individual needs of a particular site or facility, there may be several MPX-enabled server configurations, MPX Message Brokers, Root MPX Cache Agents, and Remote MPX Cache Agents serving many clients.



You can organize MPX Cache Agents hierarchically or “tiered” to support distributed teams and improve performance over slow or unreliable network connections. It also allows MPX Cache Agents to forward request “misses” and to “catch-up” with content that was missed during network or process outages. An example of tiered MPX Cache Agents is depicted in the next figure

The tiered caching capability requires the operation of a specially-configured MPX Cache Agent known as the Root MPX Cache Agent. The Root MPX Cache Agent operates directly on a server configuration’s vault. It can execute on the same computer as the server process or on another computer, as long as it has direct access to the vault. The Root MPX Cache Agent also requires access to the journal file (`CacheJournal.dat`) maintained by the MPX File Transmitter. The journal file provides the information needed by the Root MPX Cache Agent to access individual file revisions.

In the next figure, one Remote MPX Cache Agent is operating on its own computer and is being used by two clients. While all file revision transmissions are broadcast by the MPX File Transmitter, the fact that the Remote MPX Cache Agent is “tiered” to the Root MPX Cache Agent allows cache misses and catch-up requests to be forwarded to the Root MPX Cache Agent from the Remote MPX Cache Agent.



MPX Cache Agent can operate on the same computer as a client. This is especially useful for check-out intensive clients such as build applications, because the presence of a local MPX Cache Agent provides maximum performance for major check-out operations, especially those done with the `Bulk Checkout (bco)` command-line utility. See the StarTeam Cross-Platform Client online help for more information about this utility which improves check-out speeds both with and without MPX Cache Agent's help.

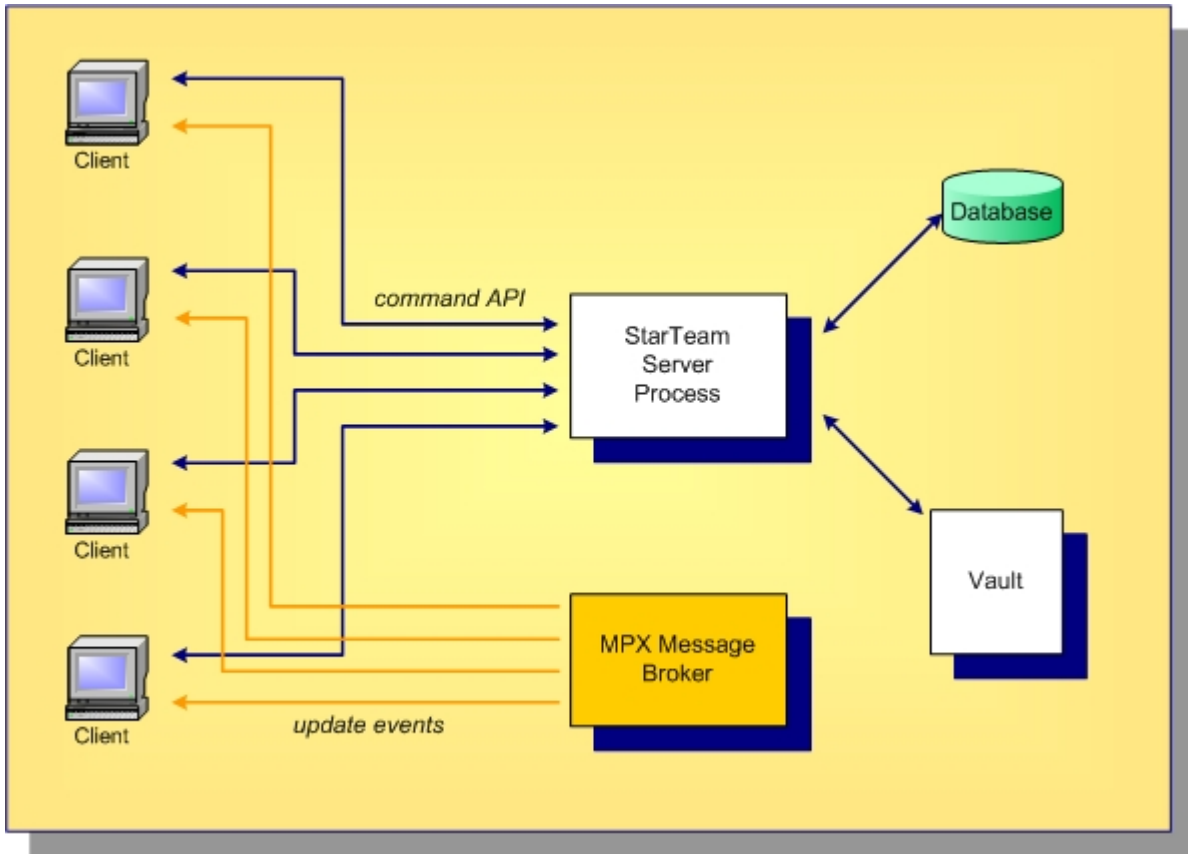
MPX Components

Like all client/server architectures, as the number of clients grows, the server could potentially become a bottleneck. In fact, the scalability of many client/server systems is entirely limited by this bottleneck. Other client/server systems address scalability by deploying multiple instances and replicating information between them to attain synchronization.

ActiveMQ MPX is a unique solution to client/server scalability. ActiveMQ MPX is a publish/subscribe messaging framework that pushes update events that contain metadata and data to clients. It is optional because it is not required for basic StarTeam functionality. However, when ActiveMQ MPX is activated, it improves StarTeam Server scalability and improves StarTeam client responsiveness.

Message Broker

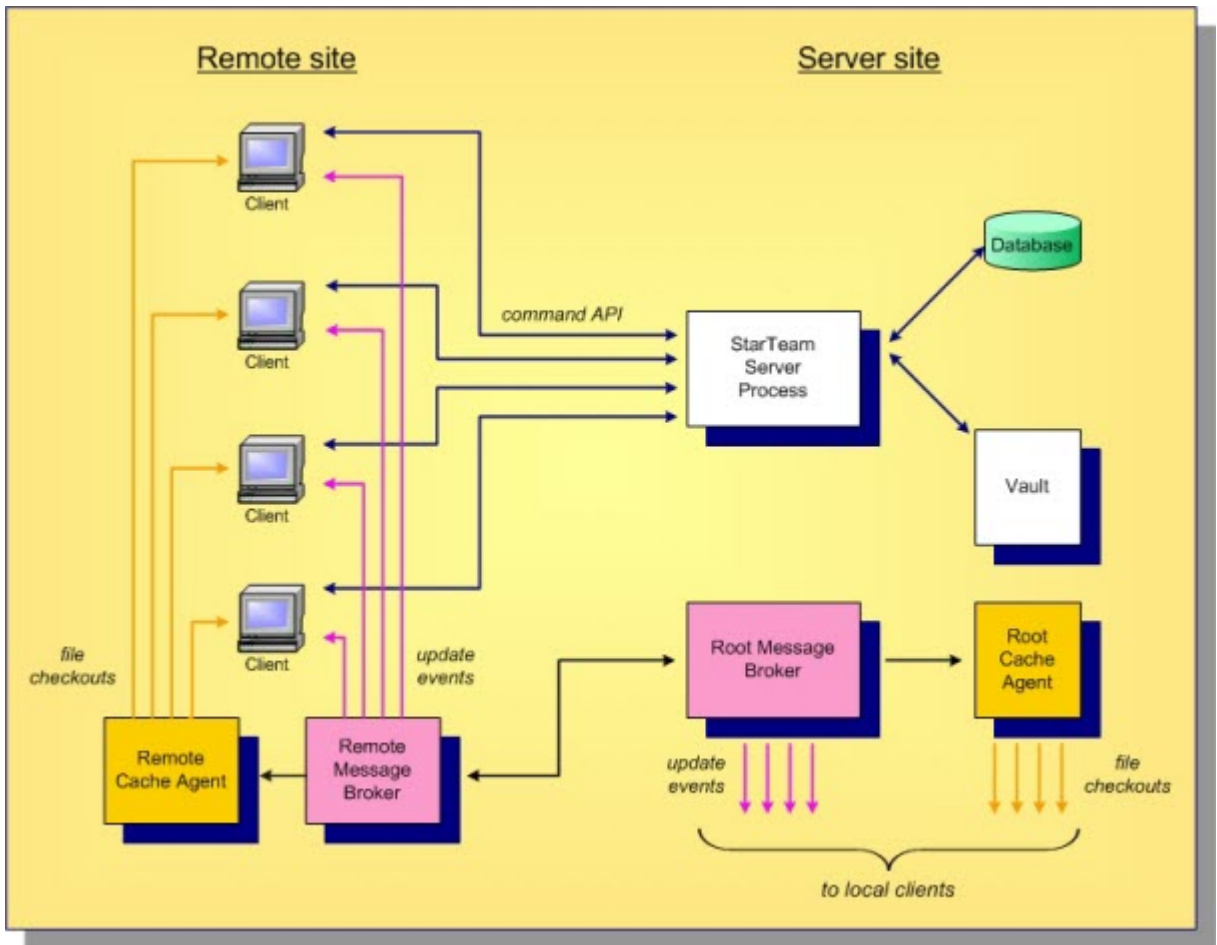
Basic ActiveMQ MPX requires the addition of a single extra component, known as the MPX Message Broker. The MPX Message Broker's role is illustrated below.



The MPX Message Broker is a messaging process that uses an event API to receive updates from the StarTeam Server process. The MPX Message Broker broadcasts encrypted messages containing updated artifacts. StarTeam clients subscribe to *subjects* and receive only messages relevant to them. By receiving updates as soon as they occur, StarTeam clients do not need to poll for updates or refresh information they have cached, significantly reducing the demand-per-client on the StarTeam Server. This improves server scalability, but it also improves client responsiveness since updates are received within seconds after they occur.

MPX Cache Agents

Messages broadcast by a MPX Message Broker benefit clients with active sessions. However, for files MPX offers an optional MPX Cache Agent process that manages its own persistent cache. MPX Cache Agents can be deployed at geographic locations, allowing clients to fetch file contents from the nearest MPX Cache Agent, preventing the need to fetch this content across a longer (and potentially slower) network connection. MPX MPX Cache Agents are illustrated below.



In this example, a Root MPX Cache Agent is deployed network-near to the StarTeam Server process. A Root MPX Cache Agent directly accesses the StarTeam vault, providing local clients with an alternate path to the vault for checking-out files. This reduces demand on the StarTeam Server, enhancing its scalability.

This example also shows a Remote Message Broker and a Remote MPX Cache Agent deployed at a remote site. Using *broker-to-broker forwarding*, each update event is forwarded once to the Remote Message Broker, which then broadcasts it to local clients. Files are streamed to the Remote MPX Cache Agent, which stores them in an encrypted private cache. StarTeam clients network-near to the Remote MPX Cache Agent can check out files at any time, leveraging the local high-speed network instead of pulling content across the WAN. This further reduces demand from the StarTeam Server while improving remote client responsiveness.

Other Options for Distributed Organizations

ActiveMQ MPX provides a unique solution for distributed teams. It leverages the benefits of a centralized server—lower total cost of ownership, better security, and simplified administration, while solving the traditional performance and scalability issues of client/server architectures. ActiveMQ MPX offers many advantages to distributed organizations:

- Any number of Message Brokers can be “chained” together (typically in a hub-and-spoke configuration) to form a “messaging cloud” that scales to any size organization. Message Broker limits can be configured to arbitrary values based on available resources such as file handles.
- Any number of MPX Cache Agents can be distributed globally. Clients can be configured to automatically locate and use the network-nearest MPX Cache Agent, or they can choose a specific MPX Cache Agent.
- MPX Cache Agents use *push caching* in which content is broadcast and stored by MPX Cache Agents as soon as it is created. This makes caches more effective than traditional “pull through” caching, in which every initial request results in a “cache miss”.

- MPX Cache Agents use advanced synchronization techniques that improve their effectiveness such as *pre-charging, tiering, request forwarding, and automatic catch-up.*

Component Descriptions

This section provides a short description for each ActiveMQ MPX component.

StarTeam Server

A StarTeam Server can support a number of StarTeam Server configurations, any or all of which can be MPX-enabled. An MPX-enabled server configuration initiates both the MPX Event Transmitter and the MPX File Transmitter. It notifies the MPX Event Transmitter each time a subscribed event occurs, and sends it relevant details about the event.

MPX Event Transmitter

The MPX Event Transmitter broadcasts events of interest to clients. The MPX Event Transmitter formats the event information it receives into XML messages, encrypts them, and publishes them to a Message Broker. Messages are assigned topics so that they can be distributed to clients interested in the accompanying content (server, item type, event type, and so on). The MPX Event Transmitter is installed when you install StarTeam Server.

MPX File Transmitter

The MPX File Transmitter broadcasts file contents and object properties to one or more Remote MPX Cache Agents by means of a Message Broker. Like the MPX Event Transmitter, the MPX File Transmitter is installed when you install the StarTeam Server.

Message Broker

The Message Broker is a publish/subscribe messaging engine that broadcasts messages to subscriber components on a topic basis. It is a stand-alone process that can run on a separate computer to offload network processing overhead in high-volume environments. The Message Broker broadcasts messages to each of its recipients using TCP/IP (unicast) messaging.

The Message Broker receives encrypted XML messages from the MPX Event Transmitter or encrypted content messages from the MPX File Transmitter, and forwards them to the appropriate clients. Information is sent from a Message Broker directly to clients that have connected to that Message Broker through a unicast (TCP/IP) connection profile.

MPX Cache Agent

MPX Cache Agent adds persistent file/object caching. Each MPX-enabled server configuration can have one Root MPX Cache Agent. One or more Remote MPX Cache Agents can be distributed throughout the enterprise.

A Root MPX Cache Agent operates directly on the server configuration's vault.

A Root MPX Cache Agent handles requests forwarded from Remote MPX Cache Agents for missing files or objects and provides "catch-up" assistance for Remote Caches after network or process outages.

Cache Agent-aware StarTeam clients can fetch files or objects from any available MPX Cache Agent.

By using "network-near" MPX Cache Agents, clients can improve file check-out and object fetch performance and reduce their demands on the StarTeam Server. This frees server resources for additional tasks and users.

StarTeam Clients - Event Transmission

When a client connects to an MPX-enabled server configuration, a connection profile determines which Message Broker the client uses to receive event messages.

Clients benefit from event messages through an enhanced internal cache. This cache subscribes to specific caching message topics, keeping its cached objects up-to-date with respect to the projects and views that the client uses. As a result, several types of object fetching (most notably item refresh) no longer require round-trips to the server. The cache is internal so message subscriptions are handled in the client.

The client cache has no persistence mechanism, and cache contents are not shared among multiple client processes. However, while the client is running, the cache remains updated with changes made to the client's open views, thereby speeding-up its operation.

A client session provides the keys required for performing MPX functions and ensures that each access is verified for applicable security context.

StarTeam Clients - File Transmission

A logged-on user can use the StarTeam Cross-Platform Client, an IDE based on StarTeam Cross-Platform Client or .NET components, or the `Bulk CheckOut (bco)` command-line utility to retrieve files from a MPX Cache Agent. The MPX Cache Agent's file/object caching is independent of client processes because the MPX Cache Agent operates as a separate process. Consequently, a client can fetch files that were broadcast while it was not operational.

ActiveMQ MPX Security

The ActiveMQ MPX security features include:

- Data encryption.
- User authentication and access rights.

Each of these features is described in greater detail in the following sections.

Data Encryption

A client receives data from an MPX-enabled server configuration over one of two paths:

- Directly from the server
- Indirectly from transmitters and MPX Cache Agents through a MPX Message Broker

The encryption level for data sent directly from the server is specified on the **Server Properties** dialog box for each individual server configuration. It is possible to have no encryption set for this data path. See the StarTeam Cross-Platform Client online help for more information on setting encryption levels for a server configuration.

All data sent by the transmitters or MPX Cache Agents is encrypted. Each MPX Event Transmitter has its own encryption key. When the server configuration starts a MPX Event Transmitter, it creates a unique encryption key for that instance of the MPX Event Transmitter. When a client opens a project, the server configuration sends the client the MPX Event Transmitter encryption key directly. The client will have one encryption key for each MPX-enabled server configuration it is accessing.

All files and objects sent by the MPX File Transmitter are encrypted. The content is stored in encrypted format by MPX Cache Agents and decrypted only "at the last moment" within the client process.

User Authentication and Access Rights

As users log on to a server configuration, they are identified individually by their user names and as members of the groups to which they belong. This information is stored as an access token for each user.

Based on a user's access rights, the server configuration determines which objects a user can see and which operations that user can perform on those objects.

The caching module in the client enforces the same user access rights set. When a client receives a message from a Message Broker, it verifies whether the user is authorized to view the data in the message. If the user has the necessary access rights, the message is stored in the client cache. Otherwise, that object will not be cached.

In a StarTeam client, you can control detailed access rights for a file: the ability to see the file, see history, check-out, check-in, and so on. For example, you can give someone the "see item and its properties" right but deny the "check-out" right.

However, with the MPX Cache Agent, granting someone the "see item and its properties" right implicitly virtually grants them a "MPX Cache Agent check-out" right. This is because the client can get a file's MD5, which is all that is needed to request a MPX Cache Agent check-out. For environments in which this difference in security "interpretation" matters, you should not deploy MPX Cache Agent or deny the "see item and its properties" right for users who should not check-out the corresponding files.

Installation

When installing MPX components, StarTeam Server must be installed first. After you have installed StarTeam Server, you can install the other components in any order.

The following is a recommendation for installing MPX components:

1. Install StarTeam Server. The MPX Event Transmitter and MPX File Transmitter are installed automatically with StarTeam Server.
2. Install the MPX Message Broker. You can run multiple instances of the MPX Message Broker.
3. Install the Root MPX Cache Agent. You need to install it only once per machine, even when that machine has more than one server configuration. You run multiple instances to support multiple StarTeam Server configurations. Each server configuration must have its own root cache agent instance.
4. Install the Remote MPX Cache Agent. This is the same installer as the Root MPX Cache Agent. You can run one or more copies on remote machines.

For complete details about installing MPX components, refer to the *StarTeam Installation Guide*.

Component Configuration

During installation, the installers pre-configure each component. However, it is recommended that you review the appropriate configuration files outlined in this section prior to starting ActiveMQ MPX.

You can configure your MPX components incrementally and in any order. For example, you could install StarTeam Server and the MPX Message Broker only, configure and start them and later add a Root MPX Cache Agent, and Remote MPX Cache Agents.

The information in this section provides an overview of some of the configuration settings that you need to be aware of when setting up your MPX components:

Review the Root MPX Cache Agent archive path

If using a Root MPX Cache Agent on a different machine than the StarTeam Server, you should update the "Root MPX Cache Agent archive path" for each hive to reflect the path with which the Root MPX Cache Agent sees that hive. You should use UNC paths on Windows because mapped drives usually do not work with services. You can edit this property in the **Hive Properties** dialog box of the **Server Administration** tool (**Tools > Administration > Hive Manager**).

MPX Transmitters

Each server configuration must have appropriately edited `MPXEventTransmitter.xml` and `MPXFileTransmitter.xml` files in a folder named `<configuration repository path>/EventServices/`. Normally these files are copied automatically from the template files (`ActiveMQEventTransmitterTemplate.xml` and `MPXFileTransmitterTemplate.xml`), which are installed at `server_installation_path/Event services/`.

You typically do not need to edit the `MPXFileTransmitter.xml` file. Its presence in the `<configuration repository path>/EventServices/` folder enables the file transmitter.

You should review each `MPXEventTransmitter.xml` file to ensure that the `<server_names>` property of each `<Profile>` points to the appropriate MPX Message Broker. Other edits such as adding more profiles may also be appropriate.

Initially, you must edit these files manually using a text editor such as Notepad. You can edit the `MPXEventTransmitter.xml` file using the **Server Administration** tool only after the server configuration is running and the MPX components have successfully started.

During your initial edits of `MPXEventTransmitter.xml`, take care to not introduce syntax errors or else the file will not load properly during the server configuration's start up. You can open the file in a browser to quickly check its syntax.

MPX Message Broker

Review the file `ActiveMQMessageBroker.ini` and edit it (using a text editor) as needed. You should use the "conn_names" property to make the service listen on a specific IP address and/or port. Use the "server_names" property to connect to other Message Brokers (to form a "cloud").

Root MPX Cache Agent

You must typically edit the configuration file. Use a text editor such as Notepad for editing this file.

Properties

The property `<ServerConfigsFile>` must point to the full path name of the appropriate `starteam-server-configs.xml` file (using a UNC path if necessary), and `<ConfigName>` must define the appropriate StarTeam configuration. Alternatively, if the root MPX Cache Agent will not use object caching, the property `<RootRepositoryPath>` must point to the repository path of the StarTeam Server configuration that it will track.

You should set the `<server_names>` property to the MPX Message Broker that the Root MPX Cache Agent will use. You may also need to adjust the `<CachePath>` property. If running a Root MPX Cache Agent for multiple StarTeam Server configurations, you must copy the configuration file with a unique file name for each configuration and edit those files with unique `<RootRepositoryPath>`, `<RequestPort>`, and `<CachePath>` values for each server configuration.

On Windows, you must also run `CacheAgentService.exe` with the "-register" and "-name" parameters (and the correct server configuration file name) to establish a unique Root MPX Cache Agent service for each configuration.

Remote MPX Cache Agent

You must edit the configuration file for each Remote MPX Cache Agent, setting the `<server_names>`, `<CachePath>`, and `<MaxCacheSize>` properties as appropriate. You should also define a `<ContentSource>` section with the appropriate `<ServerGUID>` value for each server configuration to be tracked.

Configure Clients

Modify the StarTeam Cross-Platform Client to use ActiveMQ MPX using the **Tools > Properties**. See *Configuring Clients* in this manual.

**Root and
Remote MPX
Cache Agents**

If you want to enable object caching, edit the `<ObjectTypes>` section of the corresponding configuration files.

Dependencies - Startup Order for MPX Components

The startup order of MPX components is important:

- The Message Broker is core to everything, so you should start it first. There is no particular order to starting root and remote Message Brokers. In general, if the Message Broker used by an MPX component is not running, the MPX component fails to start. For example, if a Root MPX Cache Agent is installed as an auto-start service and uses a Message Broker on the same computer, the Root MPX Cache Agent may start more quickly than the Message Broker. In this case, the Root MPX Cache Agent fails. The fix is to make the Root MPX Cache Agent service depend on the Message Broker service.
- You should start the MPX-enabled StarTeam Server configuration next. You must start it at least once before starting the Root MPX Cache Agent to create the `CacheJournal.dat` file. The first time you start a StarTeam configuration after a MPX File Transmitter has been activated, it creates the `CacheJournal.dat` file for that server configuration. If the Message Broker it uses is not running, the server configuration starts in non-MPX mode. This means the server will work, but no MPX messages will be broadcast, and "fixing" it requires restarting the server.
- You should start the Root MPX Cache Agent(s) next. If the Message Broker it uses is not running, or if the `CacheJournal.dat` file does not exist for the server configuration it is tracking, the Root MPX Cache Agent fails to start. Once the `CacheJournal.dat` file exists, the root MPX Cache Agent no longer has a start-order dependency with the StarTeam Server configuration. It can start before or after the StarTeam Server configuration has started.
- You can start the Remote MPX Cache Agent(s) at any time. It is independent of all other MPX components except for the Message Broker to which it connects. If that Message Broker is not running, the Remote MPX Cache Agent fails to start. However, if one of the Root MPX Cache Agent or the StarTeam Server configuration it is tracking are not running, it detects them when they are started.
- The MPX-enabled StarTeam Server configuration must be running before the MPX Cache Agents or clients can access it.
- The MPX Cache Agents must be running before the clients can retrieve files from them.

Configuring MPX to use ActiveMQ MPX

This chapter explains how to modify your existing MPX Event Transmitter, MPX File Transmitter, and MPX Cache Agent configurations to ActiveMQ MPX (using *Apache ActiveMQ* technology). By default, new configurations for StarTeam Server and the MPX Cache Agent will be configured to use the ActiveMQ MPX Message Broker. To switch an existing StarTeamMPX configuration to use ActiveMQ MPX Messaging, you will need to update the following configuration files:

- `MPXEventTransmitter.xml`.
- `MPXFileTransmitter.xml`.
- The Root MPX Cache Agent configuration file used.
- The Remote MPX Cache Agent configuration files used.

1. Modify `MPXEventTransmitter.xml`:

- a) Make sure the version of the file is 2.0. You should have this tag `<StExternHandlerModule version="2.0">`.
- b) Add/update the `Name` property in the `<Handler>` section. This indicates what type of messaging will be used with this configuration. Type `ActiveMQ MPX Transmitter`, which designates the use of ActiveMQ MPX-type messaging and brokers.
- c) Update the `server_names` property in all profiles with correct addresses and ports for your MPX Message Brokers.

2. Modify `MPXFileTransmitter.xml`:

- a) Make sure the version of the file is 2.0. You should have this tag `<StExternHandlerModule version="2.0">`.
- b) Add/update the `Name` property in the `<Handler>` section. This indicates what type of messaging will be used with this configuration. Type `ActiveMQ MPX File Transmitter`, which designates the use of ActiveMQ MPX-type messaging and brokers.

3. Modify *Root Cache Agent* and *Remote Cache Agent* configuration files:

- a) Make sure the version of the file is 2.0. You should have this tag `<MPXCacheAgent version="2.0">`.
- b) In the `<MessageBroker>` section, add/update the `Name` property to `ActiveMQ MPX Transmitter`.
- c) Update the `server_names` property in all profiles with correct addresses and ports for your MPX Message Brokers.

Requirements When Using StarTeamMPX and ActiveMQ MPX Together

Adhere to the following guidelines for system configurations that use both StarTeamMPX with ActiveMQ MPX:

- If a remote MPX Cache Agent is used, install ActiveMQ MPX Message Broker and StarTeamMPX Message Broker on the remote MPX Cache Agent machine.
- MPX Cache Agents (root or remote): An MPX Cache Agent configuration can only connect to one type of Message Broker at a time. As a result:
 - You must setup separate root MPX Cache Agents for all configurations that use the ActiveMQ MPX Message Broker, and for all configurations using StarTeamMPX Broker.
 - You must setup separate remote MPX Cache Agents for all configurations that use an ActiveMQ MPX Message Broker, and for all configurations using a StarTeamMPX Message Broker.
- You must upgrade to StarTeam Agile 15.0, StarTeam Web Client 15.0, and the latest Micro Focus Connect. These applications will work with both the ActiveMQ MPX and the StarTeamMPX Message Brokers.
- We recommended that you upgrade to StarTeam Cross-Platform Client 15.0 , which can connect with MPX using any of the brokers.



Important: If using a version of the StarTeam Cross-Platform Client prior to 15.0, MPX will not work on ActiveMQ MPX configurations.

Requirements When Using ActiveMQ MPX Only

Adhere to the following guidelines for system configurations that use only ActiveMQ MPX:

- If a remote MPX Cache Agent is used, install the ActiveMQ MPX Message Broker on the remote MPX Cache Agent machine.
- All MPX Cache Agents (root or remote) must be upgraded to version 15.0 and configured to use the ActiveMQ MPX Message Broker.
- You must upgrade to StarTeam Agile 15.0, StarTeam Web Client 15.0, and the latest Micro Focus Connect. These applications will work with both ActiveMQ MPX and StarTeamMPX Message Brokers.
- We recommended that users upgrade to StarTeam Cross-Platform Client 15.0 , which can connect with MPX using any of the brokers.



Important: If you are using a version of the StarTeam Cross-Platform Client prior to 15.0, MPX will not work in this scenario.

Managing Message Brokers

The Message Broker is the messaging engine. All subscribers must use a profile for a Message Broker.

Depending on the needs of your environment, you may decide to install several Message Brokers on multiple systems. The section named “*Understanding Clouds*” will help you decide.

This chapter also provides sections on configuring Message Brokers.

You can find installation instructions and system requirements in the *StarTeam Installation Guide*.

Planning for Message Brokers

Planning considerations for the Message Broker include:

- At least one Message Broker is required for ActiveMQ MPX operation.
- When the total number of subscribers is relatively low (less than 100 users) and the overall activity is low to moderate (up to 1,000 updates per hour), a single Message Broker can be installed on the same computer as the StarTeam Server.
- A single Message Broker can fulfill the messaging requirements for multiple configurations (deployed on the same computer or on multiple computers) if the total update volume is low to moderate.
- To use the same Message Broker for multiple configurations, the connection profiles in each transmitter XML file should point to the same Message Broker (see “Managing the Transmitters”).
- Subscribers should use a network-near Message Broker if possible. The StarTeam administrator must create MPX profiles to fit the needs of each general location in a distributed group and set one of them as the default profile. Some subscribers can override the default profile and should do that to select a network-near Message Broker.

Understanding Clouds

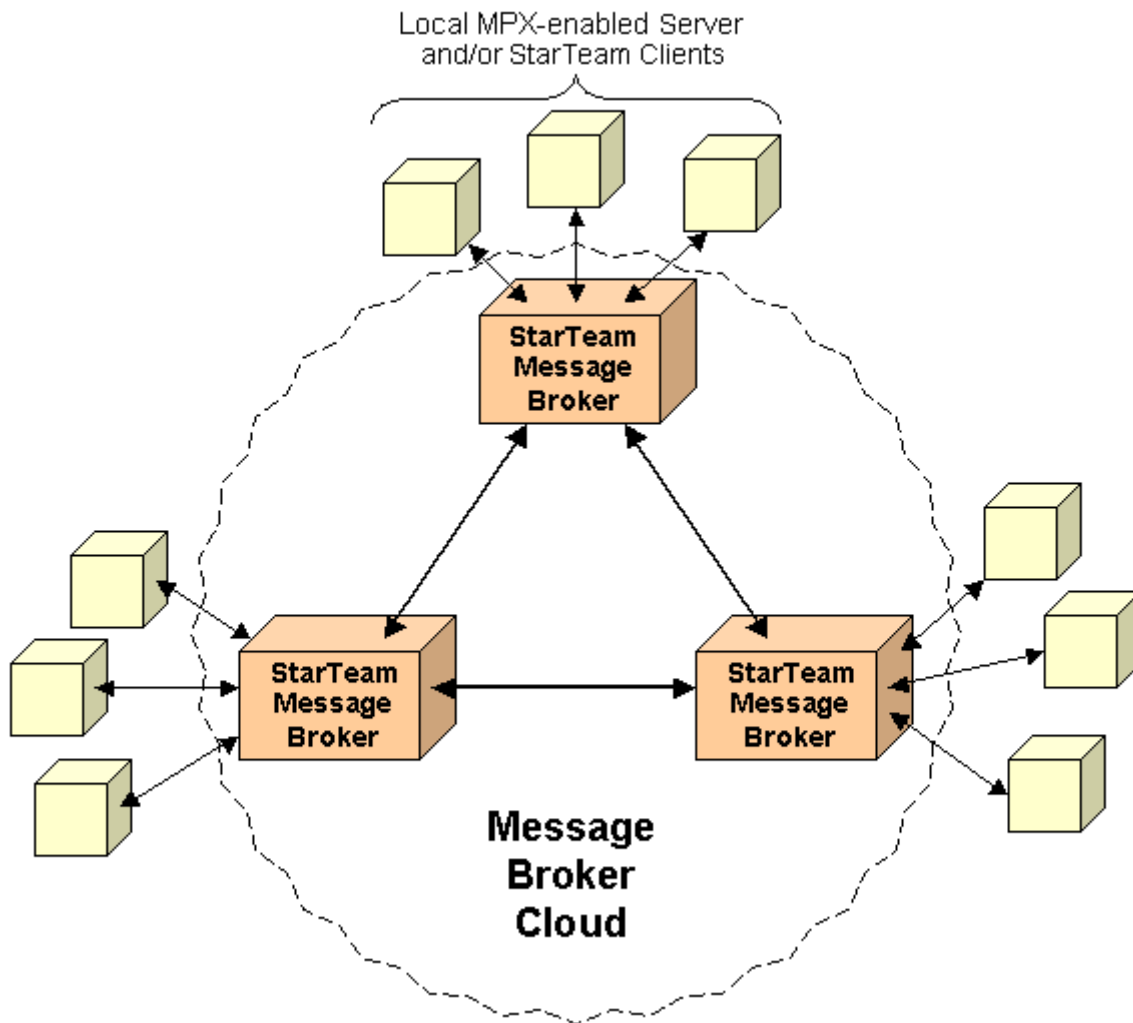
When MPX-enabled server configurations are used by clients that reside in different geographic locations, multiple Message Brokers may need to be deployed and configured to forward messages to one another. Such a configuration is called a “Message Broker cloud”.

The cloud allows a subscriber to use ActiveMQ MPX features with server configurations that reside in another geographic location.

When a Message Broker receives a forwarded message from another Message Broker, it:

- Delivers the messages to the subscribers that are connected directly to it.
- Forwards the messages to other Message Brokers that require the message.

The following shows an example of a Message Broker cloud. Some components in the ActiveMQ MPX architecture are omitted for simplicity.



Message Broker Communication

A Message Broker can establish communication with another Message Broker by:

- Initiating the communication with another Message Broker during its startup procedure.
- Being contacted by another Message Broker when that Message Broker starts.

Each Message Broker has its own `ActiveMQMessageBroker.ini` configuration file located in `C:\Program Files\Micro Focus\ActiveMQ Message Broker\conf`. The `ActiveMQMessageBroker.ini` file provides the basic configuration options required to start a broker or setup a network of brokers. ActiveMQ MPX internally uses an xml configuration file `activemq.xml` which can be edited to use the advanced features. For information on configuring `activemq.xml`, refer to <http://activemq.apache.org/xml-configuration.html>.

When a Message Broker first starts, it reads the `ActiveMQMessageBroker.ini` file and attempts to connect with each of the Message Brokers listed in `server_names` list. If any of these Message Brokers is not running or cannot be reached when contact is attempted, communication is not established with that Message Broker. However, each Message Broker will retry the connection periodically.

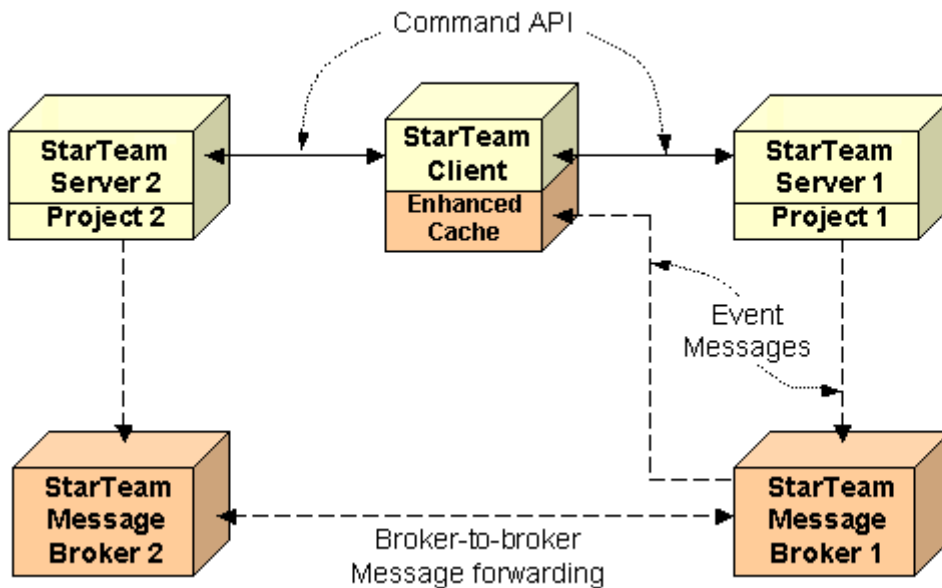
Message Routing in Message Broker Clouds

When a subscriber first connects to an MPX-enabled server configuration, it uses a connection profile to determine which Message Broker to use. If the subscriber opens another view on the same server

configuration, it reuses the same Message Broker connection. If the subscriber opens a view on a different server configuration, it will again use the same Message Broker connection if the default or selected MPX profile for that server specifies the same Message Broker. However, if the default or selected MPX profile for the new server configuration specifies a different Message Broker, the subscriber will open a new connection to it. This means that a subscriber could have multiple Message Broker connections. Each message broadcast by an MPX-enabled server configuration is automatically forwarded to every Message Broker that has a connected subscriber interested in that message.

For example, in the following figure, the client has two open projects that are located on two different MPX-enabled server configurations. When a user opens Project 1 on StarTeam Server 1, the StarTeam Cross-Platform Client chooses a connection profile for Message Broker 1.

When the user opens Project 2 on StarTeam Server 2, assume that the client has selected an MPX profile that also specifies Message Broker 1. The client sends a subscribe message to Message Broker 1 registering its interest in messages regarding Project 2. Thereafter, StarTeam Server 2 sends all messages pertaining to Project 2 to Message Broker 2, which forwards them to Message Broker 1, and finally to the client. Even if the user closes Project 1 and, therefore, its connection to StarTeam Server 1, it continues to receive messages through Message Broker 1 during the client session.



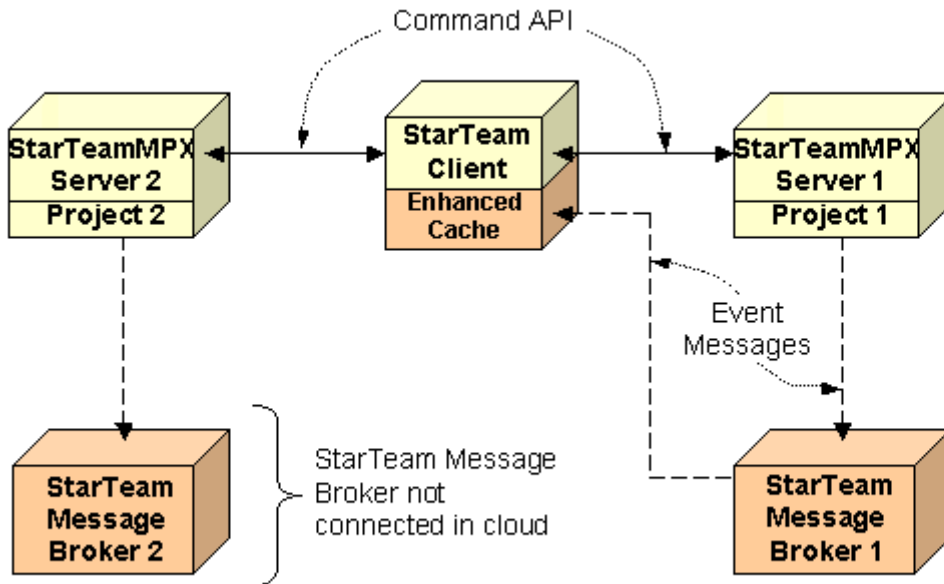
Routing in Unconnected Message Broker Clouds

In a properly configured Message Broker cloud, messages from all server configurations to which the subscriber is connected are routed to the appropriate Message Broker for that subscriber. However, in unconnected clouds, not all Message Brokers are aware of other Message Brokers and cannot forward messages appropriately.

You might have unconnected clouds because:

- The clouds are not properly configured.
- The clouds are intentionally configured this way, perhaps for increased security.

Suppose a client accesses two StarTeam Server configurations. Suppose that each server configuration uses a different Message Broker, and that these two Message Brokers are in unconnected clouds (see figure below). In this case, if the client chooses an MPX profile that specifies Message Broker 1 when it opens Project 1, the client receives messages for only the StarTeam Server configuration in the first cloud. The client treats the StarTeam Server configuration in the second cloud as if it did not support ActiveMQ MPX. The client performs all functions properly for such “disjoint” StarTeam Server configurations, but it does not receive the performance and instant notification benefits provided by ActiveMQ MPX.



Volume Considerations

In many situations, a single Message Broker is sufficient to handle the message processing of one or more MPX-enabled server configurations, depending on the average update volume of your environment.

In environments with a low to moderate number of updates (up to a few thousand updates per hour), the Message Broker can be installed on the same computer as the StarTeam Server. In this case, the Message Broker can handle the message broadcasting for MPX-enabled server configurations on the same computer, as well as for configurations operating on other computers located on the same local area network.

In moderate to high volume update environments (several thousands of updates per hour or more), you should consider operating the Message Broker on a separate computer from the StarTeam Server. This will offload CPU and network demand from the computer hosting the StarTeam Server configuration, providing an opportunity for additional distributed processing.

In some cases, you should consider installing multiple Message Brokers, each on their own computer, and connecting them together into a cloud of Message Brokers. The common scenarios in which multiple Message Brokers are advantageous are listed below.

Large number of simultaneous users

In the ActiveMQ MPX architecture, each subscriber using a unicast connection profile creates a TCP/IP connection to a Message Broker. The MPX Event Transmitter and MPX File Transmitter create additional TCP/IP connections. A single Message Broker can support between 500 and 1,000 simultaneous connections, depending upon message load. The Message Broker has a default limit of 2,000 simultaneous connections and will refuse new connection requests beyond that number. You can increase the maximum connection limit above 2,000, however, in high-volume environments, the Message Broker may not be able to adequately service all connections. In this case, you should consider installing multiple Message Brokers.

Fault-Tolerant / Load Balancing Operation

Multiple Message Brokers can be installed on separate computers and operate in parallel as "peers". The transmitters can subsequently be configured to randomly choose one Message Broker but, if it is not available, to use a second Message Broker (or a third, and so forth).

Wide Area Networks (WANs) If the network topology on which your application community operates contains multiple subnets or if the subnets are geographically distributed, you may want to install a Message Broker to serve each “local” user community and connect the Message Brokers into a cloud. This reduces message traffic over the greater network, because messages are transferred between Message Brokers only once, and then replicated locally to directly-attached subscribers.

External Users If you have users who need to access a StarTeam Server configuration over the public Internet, you may wish to operate a Message Broker that is inside the “DMZ” or completely outside of your corporate firewall. Such a Message Broker would then be connected to other internal Message Brokers, providing external users with ActiveMQ MPX functionality.

Using Message Brokers with a Firewall

In some cases, you may have users who need to access an MPX-enabled server configuration over the public Internet without using a Virtual Private Network (VPN). A common technique for providing access to ActiveMQ MPX is to install the StarTeam Server on a computer in the “DMZ” area of the corporate firewall (while hosting persistent data such as the database on a separate system behind the firewall).

Typically, that computer has two IP addresses and host names: an internal address/host name used by inside users, and an external address/host name used by outside users. In this scenario, a Message Broker can be operated on the same computer as StarTeam Server and be accessed by both internal and external users.

Alternatively, you could operate a Message Broker on a separate computer, also within the “DMZ”, and therefore also accessible to both internal and external users. However, in some cases (such as when corporate policy seeks to minimize the number of applications operating within the “DMZ”), you may wish to operate one or more internal Message Brokers behind the firewall and perhaps one Message Broker outside of the firewall. When the Message Brokers are formed into a cloud, both internal and external users receive the appropriate messages for the server configurations to which they are connected.

To connect an external Message Broker into a Message Broker cloud, it is best to modify the `ActiveMQMessageBroker.ini` file of one or more internal Message Brokers to point out to the external Message Broker. That is, modify the internal Message Broker’s `server_names` parameter to include the address of the external Message Broker. This technique is preferred because the firewall may not allow outside-in connections, thereby preventing the cloud from being formed in the opposite direction.

From a security perspective, a Message Broker can operate safely within the “DMZ” or completely external to a firewall for two reasons:

- The Message Broker is a communications server only and stores no persistent data that could become the target of a security attack.
- The cache messages are encrypted with a key dynamically generated by each MPX Event Transmitter session. Only clients who are successfully authenticated with a StarTeam Server through the logon sequence receive the key required to decipher the cache messages. Consequently, packet snooping and other eavesdropping techniques aimed at Message Broker traffic will not produce any meaningful data.

Configuring a Message Broker

Each Message Broker has an INI file that contains startup parameters:

`ActiveMQMessageBroker.ini`. This file must be located under the `<Message Broker Installation folder>/Conf`.

Microsoft Windows platforms This folder is typically at C:\Program Files\Micro Focus\ActiveMQ Message Broker\.

Linux platforms /opt/MessageBroker

Parameters that may need to be changed based on your needs are described in the next sections.

Configuring a Message Broker Cloud

If you plan to run more than one Message Broker, they probably should be configured into a cloud.

To create a Message Broker cloud:

1. If you have not already done so, install the Message Brokers. Refer to the *StarTeam Installation Guide* for instructions.
2. For each Message Broker you want to include in the cloud, open the associated `ActiveMQMessageBroker.ini` file in a text editor.
3. Add or edit the following line to list all the Message Brokers you want to include in the cloud:

```
server_names=tcp://servername:endpoint
```

where

- `servername` is the TCP/IP address or computer name where the remote Message Broker is installed.
- `endpoint` is the port number on which the remote Message Broker is listening (5101 is the default Message Broker port).

Separate multiple addresses by a comma (.). For example, the following line creates a cloud consisting of three Message Brokers: the current Message Broker and the two Message Brokers listed below.

```
server_names=tcp://ProdServer1:61616,tcp://ProdServer2:61620
```



Tip: If a Message Broker operating outside of a firewall must participate with Message Brokers behind a firewall, an inside Message Broker should be directed to establish contact with the outside Message Broker, because the firewall may prohibit the establishment of the connection in the reverse direction.

4. Save and close the file. The next time you start the Message Broker, it will establish connections with the Message Brokers listed in the `server_names` parameter.

Changing the Endpoint of a Message Broker

The endpoint (port number) of a Message Broker is specified when you install it. If you later want to configure a Message Broker to use a different endpoint, you must edit the `ActiveMQMessageBroker.ini` file.

To change the endpoint number of a Message Broker:

1. Make the change to the `ActiveMQMessageBroker.ini` file for that Message Broker:
 - a) Open the `ActiveMQMessageBroker.ini` file in a text editor.
 - b) Add or edit the following line:

```
conn_names=tcp://servername:endpoint
```

where

servername Is the TCP/IP address or name of the computer where the Message Broker runs; you can use the keyword 0.0.0.0 to designate the primary IP address of the local host independent of its host name.

endpoint Is the new endpoint you want the Message Broker to use. Default is 61616.

c) Save and close the file. Your changes will take effect the next time you start the Message Broker.

2. If this Message Broker is part of a Message Broker cloud, be sure to edit the `ActiveMQMessageBroker.ini` files associated with the other Message Brokers in the cloud.
3. For each server configuration that has one or more profiles that use this Message Broker, create or edit each profile using one of the following methods (this technique works only if the server has successfully started in MPX mode):

Use the Server Administration tool to edit the affected profiles:

1. From **Server Administration**, select the server configuration.
2. Click **Tools > Administration > Configure Server**.
3. From the **Event Handlers** tab, select `ActiveMQ MPX Transmitter` as the event handler.
4. From the **Profile** list, select an affected profile.
5. Click **Modify** to open the **Event Handlers Profile Properties** dialog box.
6. In the **Profile Properties** list, double-click `server_names`.
7. In the **Event Handler Property** dialog box, edit the appropriate endpoints in the **Property Value** field.

Your changes take effect the next time you start the server configuration.

Edit the associated MPXEventTransmitter.xml file directly

This technique should be used when the server is not currently running or not running in MPX mode.

1. Open the `MPXEventTransmitter.xml` file in a text editor.
2. Edit the `server_names` parameter in each affected unicast profile to reflect the new endpoint.
3. Save and close the file.

Your changes take effect the next time you start the server configuration.

4. For each MPX Cache Agent that accesses this Message Broker, edit the **MessageBroker** group to change the appropriate `server_names` parameter.
 - a) Open the MPX Cache Agent's XML file in a text editor.
 - b) Find the `server_names` parameter for the correct Message Broker. For example, it might look like the following: `<server_names>tcp:12.34.56.78:5101</server_names>`
 - c) Change the endpoint/port number portion of that line.

Configuring Two Message Brokers in a Fail-Over Configuration

Normally, only one Message Broker receives initial messages from a server configuration's transmitters. However, for load-balancing or in a fail-over configuration, you can use two Message Brokers as the "root" Message Broker. The following procedure describes where to install and how to configure these two Message Brokers. One can go on the server machine with the second on a network-near machine or both Message Brokers can be on network-near machines. Both Message Brokers can even be on the same machine if they used different port numbers.

The network-near machine (or machines) will not compete with StarTeam Server for memory and network bandwidth. Since the Message Broker is not especially CPU or memory bound, the system requirements for the second machine can be quite minimal. For example, a 1-CPU Pentium with ~512MB of memory is sufficient; however, 2 CPUs (or a Pentium-HT) and 1GB of memory is ideal. Note that, the Message Broker will crash if it runs out of memory.

To deploy and configure two Message Brokers on separate server machines:

1. Install two Message Brokers on separate server machines but network-near to the StarTeam Server, and set each to point to the other by adding the line:

```
server_names=tcp://other_server:61616
```

to the `ActiveMQMessageBroker.ini` file, where `other_server` is the IP address of the "other" Message Broker server machine.

2. Open the StarTeam configuration's Event Transmitter configuration file (typically `MPXEventTransmitter.xml`) in a text editor, and change the default client and server profiles to randomly select between the two Message Brokers:

```
<server_names>_random, tcp:box1:5101, tcp:box2:5101</server_names>
```

where `box1` and `box2` are the IP addresses of the two Message Broker server machines.

This setting causes both the MPX Event Transmitter and all local clients to randomly select one of the two Message Brokers. Furthermore, if one of them fails (for example, it faults, is turned off, or disconnected from the network), everyone connected to that Message Broker will fail-over to the other Message Broker. This also allows one box to be taken down for maintenance without bringing all of ActiveMQ MPX down.

Remote users should continue to use the MPX profile that points them to the network-nearest Message Broker.

Enabling Tracing for Message Brokers

If a Message Broker is failing for non-obvious reasons (for example, it isn't running or out of memory), you can turn on tracing by adding the following lines to its `<Message Broker install folder>/conf/log4j.properties` file:

```
log4j.rootLogger=DEBUG, console, logfile
```

By default, the log files are stored under `<Message Broker install folder>/data/activemq.log`. This can be changed by editing `conf/log4j.properties` file with the following line:

```
log4j.appender.logfile.file=${activemq.base}/data/activemq.log
```

You can find additional information on logging at <http://activemq.apache.org/how-do-i-change-the-logging.html>

Controlling Connections

The `Max_Client_Conns` option controls the total number of client connections that a Message Broker will allow. If unspecified, the value is 2000. Because this is a per-Message Broker setting, it does not constrain the overall cloud size. It goes in the file `ActiveMQMessageBroker.ini`.

By default, it is set to 2000:

```
Max_Client_Conns 2000
```

Managing the Transmitters

When the transmitters are automatically installed with the StarTeam Server, the Event Transmitter and File Transmitter files are placed in the StarTeam Server's installation folder. This section explains how to configure them.

Configuration-specific Transmitter XML Files

Each server configuration uses its own event and file transmitter XML files. This means that each configuration has its own set of profiles that define connection alternatives for accessing the Message Broker cloud in which the configuration operates.

During the ActiveMQ MPX installation, template files named `ActiveMQEventTransmitterTemplate.xml` and `MPXFileTransmitterTemplate.xml` are installed on the same computer as StarTeam Server. Their default locations depend on the operating system type, but typically is: `C:\Program Files\Micro Focus\StarTeam Server <version>\EventServices\`.

The `EventServices` folder is relative to the installation path of the StarTeam Server. The default installation path is shown, but another path may be used.

Initially, the `ActiveMQEventTransmitterTemplate.xml` file contains one sample unicast profile. The `File TransmitterTemplate.xml` file does not use profiles because this is already determined by the Event Transmitters.

The configuration-specific XML files are named `MPXEventTransmitter.xml` and `MPXFileTransmitter.xml`. They are usually created for you automatically based on the template files in the `<configuration repository path>/EventServices` folder. These subfolders have the same names as the server configurations.

Because the template XML files are used to create the configuration-specific XML files for new StarTeam Server configurations, a newly-edited template file becomes the configuration-specific XML file for future server configurations. The format of the template and configuration-specific XML files is the same.

You can edit any template file or configuration-specific XML file manually in a text editor. However, to edit connection profiles in a configuration-specific `MPXEventTransmitter.xml` file, it is best to use the **Server Administration** utility. See the *StarTeam Cross-Platform Client Help* for more details about configuring Event Handlers. The section entitled “*Changing the Endpoint of a Message Broker*” also offers some information about changing profile data in the `MPXEventTransmitter.xml` file.



Note: You only configure XML files for the File Transmitter if you are using the MPX Cache Agent.

Enabling Transmitters for Server Configurations

During the StarTeam Server installation, default template files (`ActiveMQEventTransmitterTemplate.xml` and `MPXFileTransmitterTemplate.xml`) are installed in the `server_installation_path/Event Services` folder. The StarTeam Server installation also creates folders for the `MPXEventTransmitter.xml` and `MPXFileTransmitter.xml` files of existing StarTeam Server configurations (if they do not already exist) and copies the template files to that folder, renaming them to `MPXEventTransmitter.xml` and `MPXFileTransmitter.xml`. The path to these files is `<configuration_repository_path>/Event Services/` folder. However, you must have an Enterprise Advantage server in order to use the MPX File Transmitter.

When you create a new server configuration using the **Server Administration** tool, the new server configuration is automatically MPX-enabled. The XML files are copied from the default location to the `<configuration_repository_path>/Event Services/` folder for the new server configuration. When creating a new server configuration using the `StarTeamServer.exe` command-line interface, the new server configuration is not automatically MPX-enabled. In this case, you must create the `<configuration_repository_path>/Event Services/` folder for the new server configuration, copy the template files from `<configuration_repository_path>/Event Services/` to the new folder, rename the template files to `MPXEventTransmitter.xml` and `MPXFileTransmitter.xml`, and edit them.

The MPX Event Transmitter enables basic MPX messaging. You can edit the default `ActiveMQEventTransmitterTemplate.xml` to include appropriate settings for all future server configurations. You can also edit each specific server configuration's file to include appropriate profiles, Message Broker settings, and so on.

The MPX File Transmitter requires the MPX Event Transmitter and controls the transmission of files and objects. You typically do not need to edit the MPX File Transmitter's `MPXFileTransmitter.xml` file. Having a copy of it in the `<configuration_repository_path>/Event Services/` folder is enough to enable the MPX File Transmitter and, in turn, the MPX Cache Agents.

Enabling MPX on Multiple StarTeam Server Configurations

When the MPX Cache Agent is installed, you are asked for the location of the server configuration's repository. A server configuration's repository is where its `CacheJournal.dat` file resides. This can be determined after the StarTeam Server is installed, the Event and File Transmitters are enabled, and the server is started. The File Transmitter will create the `CacheJournal.dat` file in the server configuration's repository folder.

To enable MPX on another server configuration, you will need to create a uniquely-named MPX Cache Agent configuration file that points to the StarTeam configuration's repository and uses a unique **Request Port**. You then need to create a MPX Cache Agent specifically for this server configuration. Here are the details:

1. Create a repository folder for your new server configuration called `C:\My`.
2. Create your new server configuration. Call it `My`, and make the repository `C:\My`.
3. Go to the MPX Cache Agent's install directory and make a copy of `RootCAConfig.xml`. Call it `MyCAConfig.xml`.
4. Open a DOS prompt and execute `'netstat -an'`. This will give you a list of ports already in use on your machine.
5. Edit `MyCAConfig.xml` and change `RootRepositoryPath` to `C:\My`.
6. Change `RequestPort` to something other than 5201 or a port already in use. Save the changed file.
7. At the DOS prompt go to the MPX Cache Agent's install folder and execute the following:

```
.\CacheAgentService -register Manual "full path to xml file" -name "name of new service" -log "log name"-verbose
```

This will create a MPX Cache Agent service for the server configuration.

8. Start the MPX Cache Agent service.

Here is an example `CacheAgentService` invocation for the "My" server configuration:

Understanding Connection Profiles

Event Transmitters use a connection profile to determine which Message Broker to use. File Transmitters do not use profiles because they use the same profile as the matching Event Transmitters.

A connection profile defines connection and usage parameters for a single Message Broker Service.

One or more connection profiles are defined in the `MPXEventTransmitter.xml` file. You can define multiple connection profiles when multiple Message Brokers have been deployed, or when multiple profiles are desired with different connection parameters.

Within `MPXEventTransmitter.xml`, one connection profile is designated as the server default profile. This profile is used by the event transmitter when it initializes.

One connection profile is also designated as the client default profile, causing that profile to be the default profile used by StarTeam clients.

In environments that use a single Message Broker, only a single unicast connection profile is needed. That profile is designated as both the server and client default, and the specified Message Broker is used by all. This scenario is viable for small to moderate user communities having less than 100 users.

Understanding the Event Transmitter

The following sections detail the Event Transmitter and its XML file.

Event Transmitter Startup

When a StarTeam Server configuration starts, it determines if it is an MPX-enabled configuration by locating a configuration-specific Event Transmitter XML file. The server attempts to load this file, identify the server default profile, and load the Event Transmitter with that profile. All startup messages and errors for the server configuration and the Event Transmitter are recorded in the configuration's server log file.

On most systems, the server's log file is located in the root folder of the server configuration's repository. See "Server Log Entries" for examples of typical startup messages.

If the configuration-specific XML file loads successfully, the Event Transmitter establishes a connection with the Message Broker identified in the server default profile.

The `Name` property in the `<Handler>` section indicates what type of messaging will be used with this configuration:

ActiveMQ MPX Transmitter Designates the use of ActiveMQ MPX-type messaging and brokers.

StarTeamMPX Transmitter Designates the use of legacy StarTeamMPX-type messaging and brokers (with SmartSockets).

If the Event Transmitter fails to connect with a Message Broker, the StarTeam Server terminates the Event Transmitter and starts the server configuration in non-MPX mode.

When a client opens a project on a server configuration, it determines whether or not the server configuration is MPX-enabled. If the configuration is MPX-enabled, the server sends the client the encryption key needed to decrypt messages from the Event Transmitter. It also sends the client the connection profiles defined in the configuration-specific Event Transmitter XML file.

Some clients have settings that allow the server's default connection profile to be overridden. If the user can select a specific profile and has done so, the client uses the profile set by the user. Otherwise, the client uses the profile marked as the client default profile.

If the client is unable to connect to the designated Message Broker, it displays an error dialog and proceeds to open the project in non-MPX mode.

Event Transmitter XML File Format

The general format of the Event Transmitter XML file is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  <StExternHandlerModule version="2.0">
    <Handler>
      <Name>ActiveMQ MPX Transmitter</Name>
      <Description>...</Description>
      <Properties>
        <ClientDefault>...</Client default>
        <ServerDefault>...</Server default>
      </Properties>
      <Profiles>
        <Profile>
          ...profile #1 information...
        </Profile>
        <Profile>
          ...profile #2 information...
        </Profile>
        ...more profiles, if any...
      </Profiles>
    </Handler>
  </StExternHandlerModule>
```

Note that in this example, the ellipses (...) denote information where some details have been left out. Some XML parameters in the Event Transmitter XML file are used to identify the Event Transmitter library to the server configuration; the presence and order of these parameters should not be modified.

The following table lists the parameters that can be modified to tailor the connection profiles within the XML file. The parameters are listed in the order in which they appear in the XML file.

Unless otherwise stated, all integer parameters have very large positive ranges. For example, all time values have a range of 1 to 2,147,483,647. In most cases, you should consider the default value the minimum setting. Use a practical upper limit based on common sense.

Table 1: Event Transmitter Parameters

Parameter	Description
ServerDefault	Defines the server default profile. Its value must match the name of a Profile within the Profile Set group in this XML file. The server default profile is used by the Event Transmitter and File Transmitter to establish a Message Broker connection. For configuration-specific XML files, you can use the Server Administration tool to set a profile as the server default.
ClientDefault	Defines the client default profile. Its value must match the name of a Profile within the XML file. The client default profile is used by clients as the default definition to establish a Message Broker connection. For configuration-specific XML files, you can use the Server Administration tool to set a profile as the client default.
Profile	Can be repeated. It is both a group for parameters and a member of the Profiles group. Each profile has a name, a brief textual description, and a set of properties. The initial profiles in the Event Transmitter XML file are specified during the transmitter's installation. In most cases, the default values are sufficient and do not need to be changed. However, if you need to add, move, or remove Message Brokers, or if you want to customize any connection profile settings, you can edit the XML file.

Parameter	Description
	<p>Name Defines the name of the profile. The name should provide a useful tip as to the purpose of the profile (such as “West-coast on-site”). If the profile is a server or client default profile, whose value should match the value of the corresponding <code>ServerDefault</code> or <code>ClientDefault</code> parameter.</p> <p>Description Provides a short textual message describing the profile. It should contain a value that helps users and administrators understand when to use the parameter (such as “Message Broker profile for users residing in the west coast office LAN”). Along with the profile name, the profile description appears in both the client and Server Administration connection profile dialogs.</p> <p>Properties This is a group inside the <code>Profile</code> group. Its inner parameters define the connection details of the profile.</p> <p>The parameters within this group specify the connection details of the profile. The most common options that can be used within the <code>Properties</code> group and their usage are described below.</p> <p>In special cases, some other options which are not described below can also be used. However, these options should only be used under the advice of our technical developer support services (http://supportline.microfocus.com).</p> <p>The parameters in the <code>Properties</code> group are listed in the next table.</p>

Table 2: Parameters for the Properties Group Inside the Profile Group

Parameter	Description
<code>server_names</code>	<p>Designates the protocol, address, and port number of one or more Message Brokers. The general syntax for this parameter is:</p> <pre>tcp:servername:endpoint</pre> <p>where</p> <p>servername The TCP/IP address or computer name of the computer running a Message Broker.</p> <p>endpoint The port number on which the Message Broker accepts connections. The default endpoint for a Message Broker connection is 61616.</p> <p>Examples:</p> <p>For a Message Broker running on a computer with the TCP/IP address of 12.34.56.78 and port number 5320, the <code><server_names></code> value would be:</p> <pre><server_names>tcp:12.34.56.78:5320</server_names></pre> <p>Any process using the profile communicates with a single Message Broker. However, you can list multiple Message Brokers (each separated by a comma) to provide alternatives:</p> <pre><server_names>tcp:HostA:61616,tcp:HostB:5101</server_names></pre> <p>The Event Transmitter and any clients using a connection profile with this <code><server_names></code> value will first attempt to connect to the Message Broker on the computer named <code>HostA</code>, using port 61616. If a connection to that Message Broker is unsuccessful, the process will attempt to connect to the Message Broker on <code>HostB</code> using port number 61616.</p>

Parameter	Description
	<p><code>_random</code></p> <p>The <code>_random</code> option is used with <code><server_names></code>. If multiple Message Brokers are available, the Event Transmitter can be instructed to randomly choose from the available Message Broker pool. This is done by adding the keyword <code>_random</code>, followed by a comma, before the list of pooled Message Brokers.</p> <p>For example:</p> <pre><code>_random, tcp:HostA:61616, tcp:HostB:61616, tcp:HostC:61616</code></pre> <p>The benefit of this option is not realized by the Event Transmitter, since it creates a single TCP/IP connection to a single Message Broker. Rather, the benefit comes when the MPX profile is used by many StarTeam clients.</p> <p>Consequently, as StarTeam clients connect and execute the same <code><server_names></code> value using the <code>_random</code> option, their Message Broker connections will be distributed among the available Message Brokers, causing automatic load balancing to take place. Note that when the <code>_random</code> option is used, if a selected Message Broker is unavailable, another Message Broker is randomly selected, until an available Message Broker is found.</p>
<code>project</code>	<p>Specifies a publish/subscribe “universe” name. The default value is <code>Starbase</code> and should not be changed.</p>
<code>enable_control_msgs</code>	<p>The default is <code>echo</code>.</p> <p>Specifies the types of control messages that the ActiveMQ MPX process (and application clients) will honor. Only the <code>echo</code> control message should normally be enabled, so this value defaults to <code>echo</code> and normally should not be modified.</p>
<code>server_keep_alive_timeout</code>	<p>integer; number of seconds. The default is 30.</p> <p>Both the Event Transmitter and application clients send an occasional “keep alive” message (analogous to a ping) to the Message Broker to ensure that it is still responding. This parameter specifies the time (in seconds) during which the keep alive response must be received. If the keep alive response is not received within the specified amount of time, the Message Broker connection is severed, and a reconnect sequence is initiated.</p> <p>Minimum value is zero, which disables the feature.</p>
<code>server_read_timeout</code>	<p>integer; number of seconds. The default is 30.</p> <p>Specifies the time (in seconds) that the Event Transmitter or client using this profile will wait for a response when performing a read operation. If no data is received from the Message Broker within the specified amount of time, a timeout occurs, and a keep alive operation is performed.</p> <p>Minimum value is zero, which disables the feature.</p> <p>Note: The <code>client_read_timeout</code> option in the <code>STMessageBroker68.ini</code> file should be set higher than the <code>server_read_timeout</code> option in all profiles in the <code>MPXEventTransmitter.xml</code> file. The <code>client_read_timeout</code> option by default is 45 seconds. This causes the clients to drive keep-alive logic, freeing-up processing from the Message Brokers.</p>
<code>server_write_timeout</code>	<p>integer; number of seconds. The default is 30.</p> <p>Specifies the time (in seconds) that the Event Transmitter or client will wait on a write operation. If a write request is not accepted by the Message Broker within the specified amount of time, a write timeout occurs. Unlike read timeouts, which trigger a keep alive sequence, write timeouts cause the Message Broker connection to be immediately severed, followed by a reconnect sequence.</p> <p>Minimum value is zero, which disables the feature.</p>

Parameter	Description
server_start_max_tries	<p>Integer. The default is 1.</p> <p>Specifies how many times the Event Transmitter or application client will traverse the <code>server_names</code> list during a connect sequence, before giving up and deciding that no messaging service is available.</p> <p>Minimum value is zero, which disables the feature.</p>
server_start_delay	<p>Integer; number of seconds. The default is 10.</p> <p>Specifies how long (in seconds) to wait between traversals of the <code>server_names</code> list. If an attempt to reconnect to a Message Broker fails, the Event Transmitter or application client will wait for up to the specified number of seconds before attempting the reconnect again.</p> <p>Minimum value is zero, which disables the feature.</p>

```

<Profile>
  <Name>Off-site</Name>
  <Description>The Message Broker profile for clients off-site.</Description>
  <Properties>
    <server_write_timeout>30</server_write_timeout>
    <server_names>tcp:123.45.6.78:61616</server_names>
    <socket_connect_timeout>10</socket_connect_timeout>
    <server_start_delay>10</server_start_delay>
    <enable_control_msgs>echo</enable_control_msgs>
    <server_max_reconnect_delay>10</server_max_reconnect_delay>
    <project>Starbase</project>
    <server_start_max_tries>1</server_start_max_tries>

    <server_keep_alive_timeout>30</server_keep_alive_timeout>
  </Properties>
</Profile>

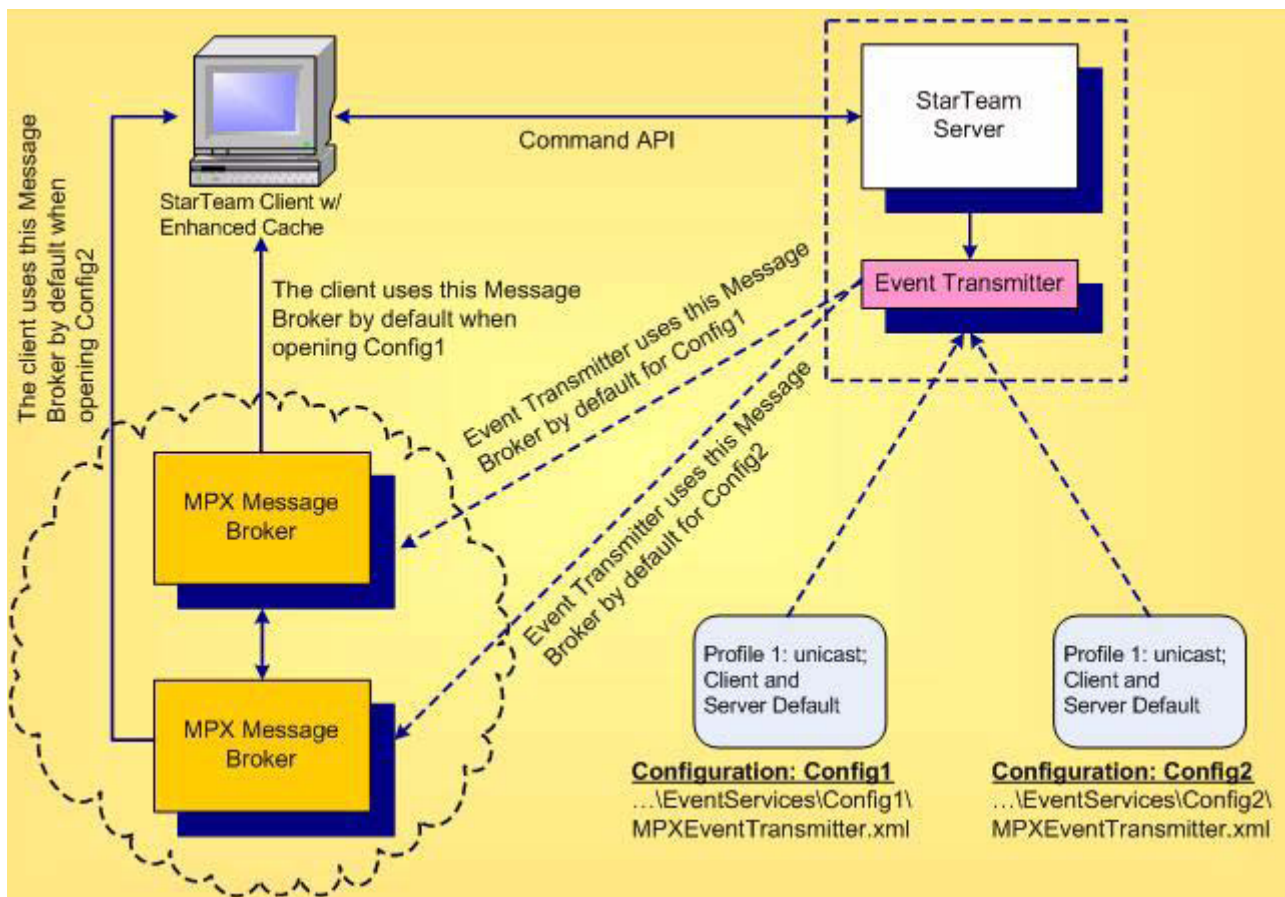
```

If you manually edit an Event Transmitter configuration-specific XML file, the changes will take effect after you save the updated file and restart the corresponding configuration. If you use the **Server Administration** tool, changes go into effect immediately for all new client connections.

If you edit an Event Transmitter template XML file, you can only edit it manually. The updated file will be used for new configurations that you create using the **Server Administration** tool.


Using Profiles with Multiple Connections

In the following figure, a sample StarTeam Server is shown with two MPX-enabled server configurations (`Config1` and `Config2`). Each configuration has its own `MPXEventTransmitter.xml` file. There are two Message Brokers: one is the default for `Config1` and the other is the default for `Config2`. Both XML files could define a profile for each Message Broker, one as the client and server default, and the other as an optional profile.



When an Event Transmitter for a specific configuration is successfully initialized, one profile is designated as the default client profile. StarTeam clients can open projects in MPX mode using that default profile or by choosing an alternate connection profile:

- When a client first logs onto a server configuration, it queries the server to determine if it is operating in MPX mode. If the server configuration is operating in MPX mode, it returns the list of connection profiles defined in its Event Transmitter XML file.
- If the user has locally chosen a specific connection profile to use for the configuration, the details of the chosen profile are loaded. Otherwise, the details of the client default profile are loaded.
- The client uses the chosen connection profile to connect to the corresponding Message Broker. If a connection cannot be established, the client displays an error message, and continues accessing the configuration as if it were not MPX-enabled.
- When a Message Broker connection is established and a new project is opened on a different, MPX-enabled server configuration, the client opens a new connection to the Message Broker specified in the default or selected profile for that server.

 **Note:** The client can also define custom connection properties that are stored locally on the client workstation. Client configuration options are described in “Configuring Clients.”

Understanding the File Transmitter

The File Transmitter obtains information about new and updated Native-II files and cacheable database objects by scanning the vault cache when it first runs and from events it receives from the StarTeam Server. It logs this information into the journal file, `CacheJournal.dat`. It periodically trims the `CacheJournal.dat` file to keep the file’s size manageable.

Root MPX Cache Agent uses this information and broadcasts files directly from the Native-II Vault.

File Transmitter Startup

The first time a server configuration uses the File Transmitter, the File Transmitter generates a new `CacheJournal.dat` file. File content transmission begins as soon as new file revisions are checked into the StarTeam Server. If any errors occur, they are reported in the server's log file.

File Transmitter XML File Format

The general format of the File Transmitter XML file is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<StExternHandlerModule version="2.0">
  <Handler>
    <Name> ActiveMQ MPX File Transmitter</Name>
    <Description>File transmitter for the MPX Cache Agent. (This event
handler does not use profiles.)</Description>
    <Properties></Properties>
    <Profiles />
  </Handler>
</StExternHandlerModule>
```

Note that in this example, the ellipses (...) denote information where some details have been left out. All File Transmitter-specific properties are optional. By default, the File Transmitter gets its mandatory parameters from the server.

The following optional properties should rarely need to be overridden.

Parameter	Description
JournalPath	Specifies the location of the cache journal file, which is created and maintained by the File Transmitter. The default is 'CacheJournal.dat' in the configuration's 'repository path' folder.
MaxJournalAge	Integer; number of days. The default is 180. Specifies the maximum age in days of journal records within the cache journal file. The minimum value is 1.
JournalTrimInterval	Integer; number of hours. The default is 24. Specifies how often (in hours) the File Transmitter trims the cache journal file of records that are older than the configured age. The cache journal is trimmed of aged records when the server first starts and every trim-interval hours. This parameter ensures that the <code>CacheJournal.dat</code> file will be periodically trimmed if the server runs for a long time without restarting. Minimum value is zero, which disables the feature.
DataTransferRate	Integer; file transmission rate in kbps (kilo bits per second). The default is 256. This value should be set to a rate that the remote MPX Cache Agent location can handle. Tests indicate even though the communication link is at 512 kbps, it is shared and therefore StarTeam would transfer at 128 kbps or lower rate. The allowable range is 64 kbps - 4096 kbps.

Managing MPX Cache Agents

MPX MPX Cache Agents provide persistent caching of StarTeam information in geographically distributed locations. By retrieving information from a network-near MPX Cache Agent, client applications can achieve better performance compared to fetching the same information from the StarTeam Server over a slower network link. Using distributed MPX Cache Agents also reduces demand on the StarTeam Server, increasing its scalability and responsiveness for other requests.

MPX MPX Cache Agents can cache file contents, thereby improving the performance of check-out commands. MPX Cache Agents also support object caching, which allows StarTeam clients to fetch properties for change requests, tasks, and other objects from a specially-configured MPX Cache Agent. As with file content caching, using a network-near MPX Cache Agent can improve the speed of the information fetch when compared to fetching the same information from a StarTeam Server over a slower network connection. Object caching can yield substantial performance improvements in many cases.

With a tier of MPX Cache Agents, there is less overall network traffic because files or objects are usually broadcast only once. They can add themselves dynamically to the “cloud”.

The Root MPX Cache Agent provides several important services:

- It supports the same fetch requests that all MPX Cache Agents provide. Because of this, it provides an alternate “pathway” to the StarTeam Server configuration’s vault and relieves the server of file fetch requests. In this capacity, the Root MPX Cache Agent can serve clients directly, although it shares vault I/O access, and – if it is running on the same computer – network access with the StarTeam Server process.
- The Root MPX Cache Agent also acts as an “upstream” MPX Cache Agent, providing “downstream” Remote MPX Cache Agents with a place to forward cache misses. This increases the “hit rate” of Remote MPX Cache Agents and improves their effectiveness.
- The Root MPX Cache Agent provides a “catch up” API, allowing Remote MPX Cache Agents to proactively fetch files that they missed while they were not running. Because the Journal is time-sequenced, a Remote MPX Cache Agent can request all content newer than the last known cache time stamp and “trickle charge” with that content before clients request it. This feature reduces cache misses and allows Remote MPX Cache Agents to be effective even over unreliable network connections.

Although a Root MPX Cache Agent can be assigned to only one server configuration, a Remote MPX Cache Agent can be assigned to several.

A Remote MPX Cache Agent can be configured to:

- Refine its subscription to a server configuration such that it receives files and objects that are created or modified only in specified projects.
- Forward request misses to either a Root MPX Cache Agent or another “upstream” MPX Cache Agent.
- Use a different upstream MPX Cache Agent for the file and object contents of each server configuration it is tracking.
- Automatically find the Root MPX Cache Agent through MPX poll messages. This allows a Remote MPX Cache Agent to be installed with a minimal configuration: all it needs is connection parameters to a Message Broker, the GUIDs of the server configurations it will track, and the names of the projects it wants to track within each server configuration. When first started, a Remote MPX Cache Agent automatically finds the Root MPX Cache Agent for each server configuration it is tracking and will trickle charge itself with the latest content that it is interested in. Even while it is trickle charging, a Remote MPX Cache Agent can be used immediately, since cache misses will be forwarded on demand.

Planning for the MPX Cache Agents

Consider the following when you plan for MPX Cache Agents:

- On Microsoft Windows, a MPX Cache Agent can run as a service or as a console application. More than one MPX Cache Agent can be run as a service on each computer if appropriate.
- A Root MPX Cache Agent can manage only one server configuration. It uses the server configuration's archives for file content. It accesses the configuration's database and manages its own local cache for object contents.
- Each MPX Cache Agent must be connected to a Message Broker.
- The Root MPX Cache Agent requires access to the vault for the one server configuration that it services. Consequently, it can be installed on the same computer as the StarTeam Server. Alternatively, if it can be installed on a separate computer to prevent the Root MPX Cache Agent from competing for CPU or network I/O with the corresponding server configuration. However, this requires it to access the archive files and the `CacheJournal.dat` through a shared network drive, so use this option only when a high-speed network file system is in place.
- Similarly, if a root MPX Cache Agent is enabled for object caching, it requires access to the server configuration's database. If it operates on a different machine than the StarTeam Server, it will need additional configuration to access the correct database.
- There is no limit to the number of MPX Cache Agents or Message Brokers that can be installed throughout an enterprise nor any limit to the number of Message Brokers within a single messaging "cloud". Keep in mind that each MPX Cache Agent requires access to a Message Broker.
- Remote MPX Cache Agents should be installed in each geographic location that can benefit from improved file check-out performance. One approach is to install a MPX Cache Agent in each network environment in which local users can access it over a high-speed LAN. (Example: Install two Remote MPX Cache Agents at headquarters, one each for the engineering and quality assurance teams, a third Remote MPX Cache Agent at the Chicago office, and a fourth at the London office.)
- Installing a Remote MPX Cache Agent on a computer dedicated to a check-out intensive application such as a build utility can be very beneficial. If that computer is sufficiently "network-near" to the StarTeam Server's computer, you could deploy a Root MPX Cache Agent on the build computer as long as that computer has access to the server configuration's vault. This reduces check-out demands on the StarTeam Server, but it doesn't reduce I/O to the vault.
- A Remote MPX Cache Agent can receive broadcasts and store files and objects from multiple server configurations. It stores all new files and/or objects for the specified server configurations or for the specified projects within the server configurations. However, each unique file and object is stored only once, regardless of the number of times it is used in different folders, projects, or servers. A file's uniqueness is determined by its contents, not its name or location. An object's uniqueness is determined by distinctive object properties.
- Cached files are stored individually within a folder tree, which has a configurable root folder. They are stored in encrypted format and decrypted only "at the last moment" within the client process.
- The maximum total size of a MPX Cache Agent's cache is configurable.
- The cache for a MPX Cache Agent does not have to be backed up and can be deleted, if necessary, when the Remote MPX Cache Agent is not running.
- Clients can be configured to use a specific MPX Cache Agent by specifying that MPX Cache Agent's host name (or IP address) and port number. Alternatively, some clients can be configured to locate an appropriate MPX Cache Agent automatically. If multiple MPX Cache Agents are available, the client automatically chooses the "network-nearest" MPX Cache Agent. This feature keeps administrative overhead to a minimum and allows the automatic detection of new MPX Cache Agents by clients.
- You can use a single Remote MPX Cache Agent without deploying a Root MPX Cache Agent. It will receive files through MPX broadcasts. If the Message Broker is running most of the time, it will receive most files and could be useful to a person whose job is building software applications. (Without a Root MPX Cache Agent there is no file-catch-up or request forwarding.)

MPX Cache Agent Operations

The following sections provide information for understanding how MPX Cache Agent operates with the Message Broker, other MPX Cache Agents, and the client.

- When a server configuration starts, so does the File Transmitter. The File Transmitter generates and maintains a `CacheJournal.dat` file for that configuration.
- When a Root MPX Cache Agent starts up for the first time, it reads the `CacheJournal.dat` file from beginning to end. This makes the MPX Cache Agent aware of the most recently added and modified files and objects and, thereby, able to return them in fetch requests. The Root MPX Cache Agent then scans the server configuration's hives for additional archive files and, if the `Precharge` and `ObjectTypes` options are set, it also scans the configuration's database for additional objects not described in the `CacheJournal.dat` file. The hive and database scans are done in the background because they may take several minutes.
- When a Remote MPX Cache Agent starts, depending on its settings, it can update its cache with data from its logical "parent" MPX Cache Agent, usually the Root MPX Cache Agent. It also adds data to its cache as it receives file and object update messages from the Message Broker.
- When a client or another MPX Cache Agent requests a file or object that the Remote MPX Cache Agent has, it returns the file or object. When asked for a file or object that it does not have but can get from a parent, it adds the new file or object to its cache and then returns it to the requesting client or MPX Cache Agent.
- If the Remote MPX Cache Agent cannot get a file or object from a parent, it sends a "miss" indicator back. After receiving "miss" indicators, clients request those files or objects from StarTeam Server in a non-MPX fetch.
- When the Remote MPX Cache Agent's total cache size exceeds the `MaxCacheSize` parameter, it removes the oldest contents in the cache. It repeats this check until the total cache size is within limits.
- Remote MPX Cache Agents provide project filtering capabilities. Specifically, you can specify a list of `Project` parameters in the `ContentSource` section of the Remote MPX Cache Agent configuration file. This list can either be explicit project names or can include a "wildcard" character, such as "*" to match partial strings.

Configuring a Root MPX Cache Agent

By default, the name of the Root MPX Cache Agent configuration file is `RootCAConfig.xml`. If you run only one instance of MPX Cache Agent on a given computer, you should probably keep the default name. Otherwise you can create multiple configuration files (whether for Root or Remote MPX Cache Agents) and run them as either services or console applications—whatever is appropriate for your environment. In general practice, you are unlikely to have more than one MPX Cache Agent per computer. See “*Running Cache Agents*” for more information.

The `<Name>` property in the `<MessageBroker>` section indicates what type of messaging will be used with this configuration:

ActiveMQ MPX Transmitter Designates the use of ActiveMQ-type messaging and brokers.

StarTeamMPX Transmitter Designates the use of legacy StarTeamMPX-type messaging and brokers (with SmartSockets).

An example of the configuration file to establish a Root MPX Cache Agent is shown below. The root element for the MPX Cache Agent configuration file must be `MPXCacheAgent`. The XML elements are summarized in *Parameters*.

```
<?xml version="1.0" ?>
<MPXCacheAgent version="2.0">
  <RootCacheAgent>
    <ServerConfigsFile>C:\Program Files\Micro Focus\StarTeam Server <version>
\starteam-server-configs.xml</ServerConfigsFile>
    <ConfigName>Sample Server Configuration</ConfigName>
    <Precharge>TipsOnly</Recharge>
  </RootCacheAgent>
  <Message broker>
    <Name>ActiveMQ MPX Transmitter</Name>
    <server_names>tcp:12.35.58.71:5101</server_names>
```



```

    <enable_control_msgs>echo</enable_control_msgs>
    <start_server_delay>10</start_server_delay>
    <socket_connect_timeout>10</socket_connect_timeout>
</Message broker>
<RequestPort>5201</Requestor>
<MaxConnections>100</MaxConnections>
<Cache types>
<Object types>
  <ObjectType>Change</ObjectType>
  <ObjectType>Requirement</ObjectType>
  <ObjectType>Task</ObjectType>
  <ObjectType>$CustomComponents$</ObjectType>
</Object types>
</Cache types>
<ListenAddresses>12.34.56.78, 21.43.65.87</Listen addresses>
<InboundAddresses>12.34.56.78, 21.43.65.87</Inbound addresses>
<MaxCatchupSize>100000000</MaxCatchupSize>
<SharePolicy>Public</Share policy>
<CachePath>C:\.MPXCacheAgent\Cache</Cyclepath>
<MaxCacheSize>1000000000</MaxCacheSize>
<MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
<MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>
</MPXCacheAgent>

```

Configuring a Remote MPX Cache Agent

A Remote MPX Cache Agent can cache content for many StarTeam Server configurations. They can be cache files and/or objects for all the projects managed by the server configuration or be set up to filter for specific projects within each server configuration.

An example of a configuration file that defines a Remote MPX Cache Agent is shown below. The absence of a `RootCacheAgent` element denotes the configuration for a Remote MPX Cache Agent.

```

<?xml version="1.0" ?>
<MPXCacheAgent version="2.0">
  <Message broker>
    <Name>ActiveMQ MPX Transmitter</Name>
    <server_names>tcp:12.34.56.78:5101</server_names>
    <enable_control_msgs>echo</enable_control_msgs>
    <start_server_delay>10</start_server_delay>
    <socket_connect_timeout>10</socket_connect_timeout>
  </Message broker>
  <RequestPort>5201</Requestor>
  <MaxConnections>100</MaxConnections>
  <Cache types>
    <Object types>
      <ObjectType>Change</ObjectType>
      <ObjectType>Requirement</ObjectType>
      <ObjectType>Task</ObjectType>
      <ObjectType>$CustomComponents$</ObjectType>
    </Object types>
  </Cache types>
  <ListenAddresses>12.34.56.78, 21.43.65.87</Listen addresses>
  <InboundAddresses>12.34.56.78, 21.43.65.87</Inbound addresses>
  <MaxCatchupSize>100000000</MaxCatchupSize>
  <SharePolicy>Public</Share policy>
  <CachePath>C:\.MPXCacheAgent\Cache</CachePath>
  <MaxCacheSize>1000000000</MaxCacheSize>
  <MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
  <MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>
  <Content source>
    <ServerGUID>be5ee3b0-c719-49c6-ala1-f493764a03f5</ServerGUID>

```

```

<Upstream cache>
  <UpstreamHost>ProdServer1</UpstreamHost>
  <UpstreamPort>1123</UpstreamPort>
</Upstream cache>
</Content source>
<Content source>
  <ServerGUID>79408139-1768-4031-9ddd-7f1b095c94e7</ServerGUID>
  <Projects>
    <Project>FelixTools</Project>
    <Project>Bank*</Project>
    <Project>Insurance*West*</Project>
  </Projects>
  <Upstream cache>
    <AutoLocate/>
  </Upstream cache>
</ContentSource>
</MPXCacheAgent>

```

Cache Agent XML Parameters

The XML elements for both Root and Remote MPX Cache Agents are summarized in the following sections. Unless otherwise stated, all integer parameters have very large positive ranges. For example, all time values have a range of 1 to 2,147,483,647. In most cases, you should consider the default value the minimum setting. Use a practical upper limit based on common sense. For example, if you set the `CacheCheckInterval` at 10,000,000, the cache will only be checked once every 4 months or so, which makes cache management ineffective.

Parameters Used by Any MPX Cache Agent

CacheCheckInterval

Integer; number of seconds. The default is 60.

The frequency with which the MPX Cache Agent compares its cache size to the configured cache limit (`MaxCacheSize`). When the total cache size exceeds the configured limit, least-recently-used files are removed from the cache until the cache size is under the configured limit.

Minimum value is zero, which disables the feature.

CachePath

Path. The default is `"/MPXCacheAgent/Cache"`.

The root folder of the MPX Cache Agent's local cache. Cached files are stored in compressed, encrypted format within subfolders of this path.

As files are requested from the Root MPX Cache Agent either by "downstream" MPX Cache Agents or by clients, they are compressed, encrypted, and stored in a folder tree rooted at the specified directory. The local cache makes secondary file access faster, and it removes I/O contention with files in the server configuration's vault.

The Root MPX Cache Agent uses the StarTeam Server configuration's cache.

CacheTypes

Specifies the type of information that will be cached by the MPX Cache Agent. If this group is not specified, the MPX Cache Agent caches file contents only.

ObjectTypes Specifies one or more object types to be cached by the MPX Cache Agent. If this group is specified with

the attribute `AllTypes="True"`, then all object types are cached by the MPX Cache Agent. Otherwise, the object types cached are defined by child `<ObjectType>` elements. If this group is not specified, no object types are cached by the MPX Cache Agent. The object types listed in the Remote MPX Cache Agent must be the same or a subset of the object types listed in the Root MPX Cache Agent.

ObjectType A valid object type such as Change, File, Folder, Requirement, Task, Topic or the name of a Custom Component created for this installation. The special entry `$CustomComponent$` can also be used.

Caching File objects means that artifact properties are cached, which is independent from caching file content.

The `$CustomComponent$` setting is used to cache all custom components, including new ones added while the server is running. This can be used instead of adding separate entries for each custom component created, which would require restarting the MPX Cache Agent.

InboundAddresses

Comma-separated list of IP addresses. By default, the MPX Cache Agent uses a list of its machine's IP addresses.

Returned by the MPX Cache Agent in "poll" responses, this list also appears as the "Inbound Addresses" value on the MPX Cache Agent's status page.

In cases where some addresses should be hidden from clients (for example, they are not routable) or the MPX Cache Agent has additional addresses that should be presented to clients (for example, an additional address assigned by a firewall's NAT), this option can be used to control exactly what list is presented to clients.

InitialRequestThreads

Integer; number of connections. The default is 10. The range is from 1 to `MaxConnections`.

The initial number of request handler threads launched when the MPX Cache Agent starts. Additional request handler threads are launched, up to `MaxConnections`, as needed when all current threads are dedicated to active connections.

ListenAddresses

Comma-separated list of IP addresses. By default, the MPX Cache Agent binds to the "system" address (`IPADDR_ANY`), which allows it to receive connections from all available physical (for example, NIC) and logical (for example, VPN) IP addresses.

This list allows you to control the exact set of IP addresses with which the MPX Cache Agent will listen for inbound connections. This list must contain only IP addresses that are valid for the current host and/or the loopback address (127.0.0.1) that allows connections only from the local host.

Unless the `InboundAddresses` value is specified, the addresses provided in this option are used in poll requests and displayed in the "Inbound Addresses" value on the MPX Cache Agent's status page.

MaxCacheSize

Integer; number of bytes. The default is 1000000000. The range is 0 to approximately 8 exabytes.

The maximum size of the MPX Cache Agent's cache in bytes. If this value is zero, the cache size will not be constrained. Otherwise, files are periodically deleted on a least-recently-used basis to maintain the specified size.

The actual total size of the cache may rise above this value momentarily while new files are received.

MaxCatchupSize

Integer; number of bytes. The default is 100MB. The range is 0 to 263-1.

For a Root MPX Cache Agent, this parameter constrains the maximum number of files returned in a catch-up request. For a Remote MPX Cache Agent, this parameter constrains the maximum number of files requested in a catch-up request. In both cases, the value is the total size in bytes of the files in the catch-up operation. If this value is 0, the catch-up request is unconstrained. In a given catch-up operation, the smaller of the requester's and the requestee's `MaxCatchupSize` is used to constrain the operation.

MaxConnections

Integer; number of connections. The default is 100. The range is 1 to 1000.

The maximum number of simultaneous connections that the MPX Cache Agent will accept. This value controls the maximum number of request handler threads used by the MPX Cache Agent. If all request handler threads have been started and are in use when a new connection is received, it is queued until a request handler becomes available.

A larger value than the default could be used in highly concurrent environments, at a cost of more memory and potentially more demand on the MPX Cache Agent. A smaller number is generally unnecessary. The maximum is limited by OS/process issues.

MessageBroker

Required group. A group for specifying Message Broker parameters. Minimally, a MPX Cache Agent's `MessageBroker` group should contain the `<Name>` tag and a value for the `server_names` parameter.

The `<Name>` property in the `<MessageBroker>` section indicates what type of messaging will be used with this configuration:

ActiveMQ MPX Transmitter

Designates the use of ActiveMQ-type messaging and brokers.

StarTeamMPX Transmitter Designates the use of ActiveMQ-type messaging and brokers.

A Remote MPX Cache Agent uses a Message Broker connection to receive file content messages from the content source(s) that it is monitoring. It also uses the Message Broker connection if it is a “public” MPX Cache Agent that will respond to poll messages. (See *SharePolicy*.)

server_names Defines the publish/subscribe messaging service to be used by the MPX Cache Agent in the format `tcp:host:port`.

The default is the value “_node” which is equivalent to “localhost”. To use the default, the Message Broker must be on the same computer as the MPX Cache Agent.

`host` must be the name or IP address of the Message Broker computer, and `port` must be the port number with which the Message Broker is receiving connections (61616 by default). Example:
`tcp:MBServer1:5101.`

server_start_delay Integer. Number of seconds. The default is 10.

Specifies the interval between attempts to reconnect to the Message Broker.

socket_connect_timeout Integer. Number of seconds. The default is 5.

Controls how long in seconds the Event Transmitter, MPX Cache Agent, or StarTeam client waits to connect to the Message Broker before giving up. The default is good for most environments. Minimum value is zero, which disables the feature.

MemoryCacheMaxSize The maximum amount of memory in bytes that will be used for memory caching. When enabled, most-recently-used cached objects are buffered in memory. A value of 0 disables memory caching. If this value is not specified, the default value is 100MB. This value appears as **Memory Cache Max Size (bytes)** on the MPX Cache Agent’s status page.

MemoryMaxCacheObjectSize The maximum size of an object that will be cached in memory. When memory caching is enabled, only new and recently accessed objects at or below this size in bytes will be memory cached. If this value is not specified, the default value is 10KB. This value appears as **Memory Cache Max Object Size (bytes)** on the MPX Cache Agent’s status page.

RequestPort Integer; port number. The default is 5201. The range is 1 to 65535.

The port number with which the MPX Cache Agent receives requests. The port number cannot be in use by any other process on the same host.

RequestReadTimeout

Integer; number of seconds. The default is 30.

Specifies the time which a MPX Cache Agent will wait to read the next request from a client before passively closing the corresponding connection. When the connection is closed, the request handler thread is freed-up to service other connections.

Minimum value is zero, which disables the feature.

SharePolicy

Public or Private. The default is Public.

Indicates whether or not this MPX Cache Agent advertises the server GUIDs for which it is caching data. A cache-aware MPX client can broadcast a "poll" request to look for MPX Cache Agents that are caching data for a specific StarTeam Server GUID. Public MPX Cache Agents will respond to such requests when they are caching the requested server's content, but private MPX Cache Agents will not.

Parameters Used by Remote MPX Cache Agent

CatchupCheckInterval Integer; number of seconds. The default is 300.

The interval in which a remote MPX Cache Agent will check to see if any of its cache sources require catch-up because the Message Broker connection was lost. When catch-up is required, catch-up cycles continue to be performed until normal Message Broker connectivity has been resumed.

Minimum value is 1.

ContentSource

Each `ContentSource` group specifies a StarTeam Server configuration that the Remote MPX Cache Agent will monitor for new file content via MPX messages. Optionally, this group specifies an "upstream" MPX Cache Agent that will be contacted for catch-up and forwarding requests. A Remote MPX Cache Agent can have one or more `ContentSource` groups.

It can contain the following parameters:

ServerGUID

This value must be specified within each `ContentSource` group. It must be the GUID of the StarTeam Server that this remote MPX Cache Agent will track.

It is used to establish the publish/subscribe "subjects" used to monitor new file content. If the GUID is specified incorrectly, the MPX Cache Agent will not receive new file content.

To locate the GUID for a specific server configuration:

1. From the **Server Administration** tool, select the server.
2. Click the **Properties** icon on the toolbar or click **Server > Server Properties**. These actions display the **Properties** dialog box.

3. Copy the GUID from the server configuration's **Properties** dialog box and paste it into the .xml file.

Projects

If specified, this group value indicates that not all content for the corresponding StarTeam Server is to be tracked. Instead, only content for each `Project` parameter within the `Projects` group will be tracked and stored. It contains one or more `Project` parameters.

Project

Each `Project` parameter specifies one project name or name pattern. All files checked into a project whose name matches the specified name or pattern are cached by the MPX Cache Agent. A pattern is a name that contains one or more asterisk (*) wildcard characters.

For caching purposes, each file revision "belongs" to the project it is checked in to. In practice, a file could be shared among multiple projects and a given check-in may cause the corresponding file revision to appear in multiple projects. Because the file content is "broadcast" with the project name it was checked into, only Remote MPX Cache Agents tracking that project will store the file. For example, suppose a file is shared between projects P1 and P2, and a new revision of the file is checked into project P1. Only Remote MPX Cache Agents tracking project P1 store the new file revision. A Remote MPX Cache Agent tracking only project P2 will not receive the broadcast. However, "pull through" caching, explained more below, allows the Remote MPX Cache Agent that is tracking only project P2 to obtain the file revision anyway.

When a Remote MPX Cache Agent is configured to track specific projects, it still accepts requests for any file. If a requested file is not in its local cache, a Remote MPX Cache Agent forwards the request to the appropriate Root MPX Cache Agent and stores it locally for future requests, regardless of which projects the file belongs to. In other words, tracking-by-project limits files visible to Remote MPX Cache Agents by "push" caching, but it does not limit the files a Remote MPX Cache Agent can store via "pull through" (request forwarding) caching.

UpstreamCache

This group specifies that an upstream MPX Cache Agent will be contacted for catch-up and forwarding requests for content related to the corresponding StarTeam Server. The upstream MPX Cache Agent can be automatically located or explicitly configured as specified by the group's parameters: `AutoLocate`, `UpstreamHost`, and `UpstreamPort`.

If a `ContentSource` parameter does not contain an `UpstreamCache` parameter, catch-up and miss

forwarding will not occur for the corresponding server configuration. When used, `UpstreamCache` must contain either an `UpstreamHost` or an `AutoLocate` parameter. The two are mutually exclusive. `UpstreamHost` (along with `UpstreamPort`) explicitly specifies the upstream MPX Cache Agent, while `AutoLocate` requests polling.

AutoLocate	Indicates that the Root MPX Cache Agent for the corresponding StarTeam Server configuration is to be automatically located and used as the upstream MPX Cache Agent. Remote MPX Cache Agent to continue periodic polls for a Root MPX Cache Agent until one is found. It is an empty tag and mutually exclusive with <code>UpstreamHost</code> .
UpstreamHost	Explicitly identifies the host name or IP address of the upstream MPX Cache Agent to be used for content related to the corresponding StarTeam Server. This parameter is mutually exclusive with <code>AutoLocate</code> .
UpstreamPort	Integer. The default is 5201. The range is from 1 to 65535. Only meaningful if <code>UpstreamHost</code> is specified, this parameter specifies the port number of the upstream MPX Cache Agent.

PrechargeSize	Integer. The default is <code>MaxCatchupSize</code> . Specifies the maximum size of a Remote MPX Cache Agent's first catch-up operation, which is used to precharge its cache. It defaults to the MPX Cache Agent's <code>MaxCatchupSize</code> , but it can be specified larger or smaller than that value. If this value is zero, the Remote MPX Cache Agent does not perform an initial pre-charge operation.
----------------------	---

Parameters Used by Root MPX Cache Agent

RootCacheAgent	Group for Root MPX Cache Agent parameters as defined below.
ServerConfigsFile	This option specifies the full path name of the StarTeam Server configuration file. It is required for object caching.
ConfigName	This option specifies the name of the StarTeam configuration that the root MPX Cache Agent will cache. It must be used in conjunction with the <code><ServerConfigsFile></code> option.
PreCharge	Specifies whether the root MPX Cache Agent should "pre-charge" its local cache with missing object revisions by scanning the database at start-up time. Values include: <ul style="list-style-type: none"> "None" A pre-charge is not performed. "TipsOnly" (Default). Only tip revisions are added to the local cache.

"All"	All object revisions are added to the local cache. This option is ignored if the <code><ObjectTypes></code> option is not specified.
RootRepositoryPath	Required. The full path name of the StarTeam Server configuration's repository folder. This option is needed only when <code>ServerConfigsFile</code> is not used.
RootHiveIndexPath	The full path of the StarTeam Server's Native-II hive index file. The default is <code>RootRepositoryPath/HiveIndex/hive-index.xml</code> .
RootJournalPath	The full path name of the StarTeam Server configuration's cache journal file, which is created by the File Transmitter. The default is <code>RootRepositoryPath/CacheJournal.dat</code> .

Reviewing Status and Log Information

You can review status and log information for each MPX Cache Agent in your browser by using the MPX Cache Agent's port number and the name (or IP address) for the computer on which the MPX Cache Agent is located. The status information for a Root MPX Cache Agent and a Remote MPX Cache Agent are slightly different.

To check the status of an active MPX Cache Agent:

1. Open your browser.
2. Enter the appropriate URL using the following syntax.

```
http://host:port/status
```

`host` is the MPX Cache Agent's host name or IP address.

`port` is the inbound port on which it is listening.

Similarly, you can use the syntax:

```
http://host:port/log
```

to view the MPX Cache Agent's log file from a browser.

If verbose mode was used when the MPX Cache Agent was started, you can use the following syntax:

```
http://host:port/debuglog
```

to display a more detailed log.

Using MPX Cache Agent with the Clients

The StarTeam Cross-Platform Client, IDEs based on StarTeam Cross-Platform Client, or .NET components, and the Bulk Checkout (bco) command-line utility can perform check-out operations using MPX Cache Agent.

See "Using Cache Agent from the Cross-Platform Client and IDEs" for information about using MPX Cache Agent when checking out files with this client.

See the *StarTeam Cross-Platform Client Help* for more information.

Object Caching

At the client level, the primary unit of information that we see is called an item. But “under the hood”, an item is really comprised of two parts:

- An object (or artifact) contains most of the persistent information we see in an item. The object has properties such as **Name**, **Description**, **Created By**, **Version**, and so forth.
- A view member associates a specific object with a specific view and folder. In addition to properties that identify the view, folder, and object, the view member has configuration properties such as **Branch On Change** and **Configuration Time** that affect how the object is accessed through the corresponding view.

View members are specific to a view, whereas objects can be referenced by any number of views. At the client level, view member and object properties are blended into the unified items that users see. The information associated with view members is comparatively small. In contrast, the properties of a single object revision can consist of up to 100KB of data or more. When a large number of items are fetched from the StarTeam Server, the response can be several megabytes in length. Even when a client enables compression, a large fetch request can require several seconds to a minute or more depending on network speed and latency.

When object caching is enabled, an MPX Cache Agent stores most (but not all) of the properties for configured object types. An object caching-aware client can then fetch those properties from a network-near MPX Cache Agent, resulting in faster performance compared to fetching from the StarTeam Server. Object fetching is performed by the StarTeam SDK, which combines view member properties and some object properties retrieved from the StarTeam Server with object properties fetched from a MPX Cache Agent, yielding the requested items. In other words, when object caching is enabled, there is no detectable difference to client applications except for (in many cases) greater performance.

Not all properties are cached when object caching is enabled. Some properties are view-specific (e.g., **Branch On Change**), user-specific (e.g., **Flag User List**), server-calculated (e.g., **Attachment Names**), client-calculated (e.g., **My Lock**), or modifiable without creating a new revision (e.g., **Comment**). For different reasons, these properties are not “cacheable” and therefore are not included in persistent cache objects. When needed, these properties must be either calculated by the SDK or fetched from the StarTeam Server.

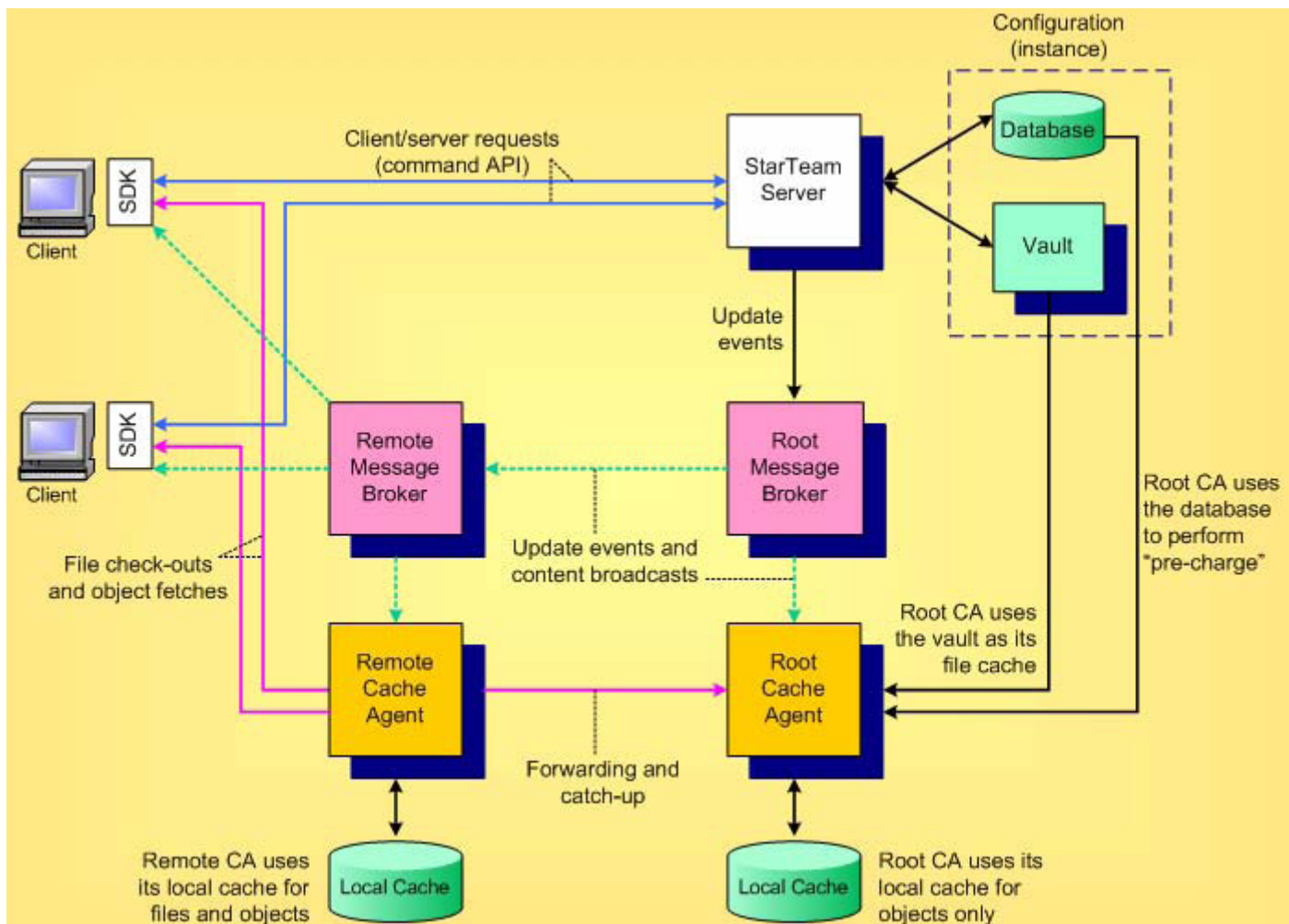
If not all object properties are cached, how effective is object caching? The vast majority of object properties, both in number and total size, are “eligible” for caching, hence object caching can be effective even though not all properties are cached. For example, for change request objects, the following properties are cached:

Addressed By, Addressed In, Addressed In View, Attachment Count, Attachment IDs, CR Number, Category, Closed On, Component, Created By, Created Time, Deleted By, Deleted Time, Description, Dot Notation, End Modified Time, Entered By, Entered On, External Reference, Fix, Last Build Tested, Modified By, Modified Time, Object ID, Parent Branch Revision, Parent ID, Parent Revision, Platform, Priority, Resolved On, Responsibility, Revision Flags, Root Object ID, Severity, Status, Synopsis, Test Command, Type, Verified On, Version, View, Work Around ...plus all custom (user-defined) properties.

As with file caching, object caching uses encryption and other security measures so that cached objects can only be retrieved by authorized users. The use of object caching does not compromise StarTeam-defined security in any way.

How Object Caching Works

To examine how object caching works, let’s examine the primary MPX components. The diagram below illustrates a typical deployment with all major MPX components.



As in all StarTeam deployments, one or more StarTeam clients communicate with a StarTeam Server using a client/server protocol called the command API. On the client side, all client/server interaction is managed by the StarTeam SDK. The StarTeam Server manages data for a single configuration (instance), which consists of a database and a vault.

To enable the basic StarTeam client/server configuration with MPX capabilities, the first component deployed is typically a root Message Broker. When the StarTeam Server is suitably configured, it sends update events to the root Message Broker which broadcasts them using publish/subscribe messaging. Typically, each remote location will deploy one remote Message Broker, which receives one copy of each event message. The remote Message Broker then relays specific update event messages to remote StarTeam clients. This “pushes” updates to clients, preventing them from refreshing or polling for new information. (Though not shown above, the root Message Broker also broadcasts update messages to network-near clients that are directly connected to it.)

The next layer of MPX functionality that can be enabled is distributed file caching. First, a root MPX Cache Agent is deployed network-near to the StarTeam Server, and a remote MPX Cache Agent is deployed in each remote location. As new file revisions are created, their content is broadcast by Message Brokers and cached locally by remote MPX Cache Agents. (A root MPX Cache Agent does not need to cache file content since it has direct access to file revisions in the configuration’s vault.) When a StarTeam client is enabled for MPX Cache Agent usage, it can check-out files from a network-near MPX Cache Agent at substantially greater speed compared to accessing the StarTeam Server over a limited-bandwidth connection.

The same MPX caching framework also supports object caching. Message Brokers broadcast new and modified objects, which are subsequently cached by both the root MPX Cache Agent and remote MPX Cache Agents. Clients that enable MPX Cache Agent access can subsequently fetch specific object

revisions from a network-near MPX Cache Agent, again at potentially greater speed. The StarTeam SDK utilizes an adaptive performance monitoring algorithm to determine when MPX Cache Agent access is beneficial. Consequently, it only uses a MPX Cache Agent for object fetching when it is faster. This algorithm adapts to changing network demands, peak load periods, and other factors so that the fastest possible fetch technique is always used.

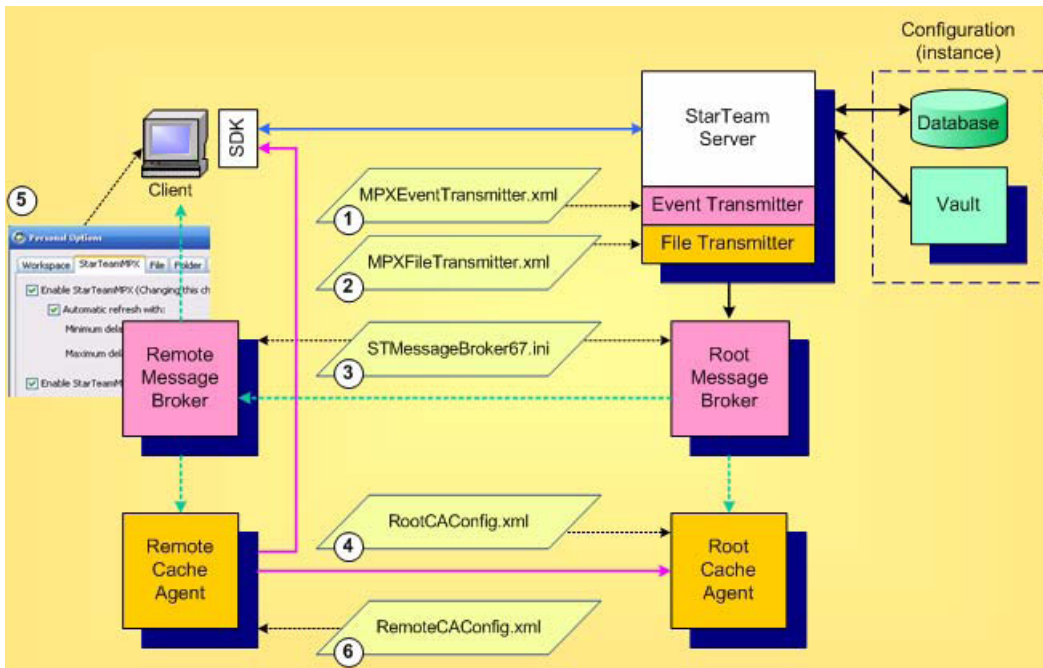
Components Needed for Object Caching

Object caching requires the following StarTeam components:

StarTeam Server	A StarTeam Server version 2009 or later with an Enterprise Advantage license is required to use object caching. (Basic MPX functionality can be used with a StarTeam Enterprise license, but the use of MPX Cache Agents require an EA license.)
Root MPX Cache Agent	One MPX MPX Cache Agent version 2009 or later must be configured as a “root” MPX Cache Agent for each StarTeam configuration that uses object caching. MPX Cache Agents always cache file content; object caching requires additional configuration. When a pre-2009 StarTeam Server is upgraded, the root MPX Cache Agent for its StarTeam configuration(s) must be upgraded to the same release at the same time.
Remote MPX Cache Agents	One or more MPX MPX Cache Agents can be configured as “remote” MPX Cache Agents in each desired location. A MPX Cache Agent version 2009 or later is required to support object caching. Pre-2009 remote MPX Cache Agents will interoperate with a 2009 or later root MPX Cache Agent, but they will only support file caching. If a 2009 or later remote MPX Cache Agent is configured to use a pre-2009 root MPX Cache Agent as its “upstream” MPX Cache Agent, it cannot be enabled for object caching and will only support file caching.
StarTeam SDK	To use object caching, a StarTeam client application must use the StarTeam SDK version 2009 or later. If an application using an older SDK communicates with a 2009 or later MPX Cache Agent, it will only be able to fetch file contents. Likewise, an application using a 2009 or later SDK will only be able to fetch file contents if it communicates with a pre-2009 MPX Cache Agent.
Message Broker	One or more MPX Message Brokers are required to use MPX functionality, including caching. Typically, one Message Broker is deployed in each location that has a MPX Cache Agent, often on the same machine. There are no inter-version requirements with Message Brokers and object caching: 2009 and previous-version Message Brokers can be mixed and will interoperate in the same messaging “cloud”.

Configuring Object Caching

The best way to enable object caching is to install and configure MPX components in a specific order. The diagram below provides an overview of the MPX components, emphasizing items that can be configured, and the order in which they should be configured.




Each of these configuration items are discussed in the following sections.

1, 2 Configuring the MPX Transmitters

When you install the StarTeam Server, two included components are the MPX transmitters. Basic MPX services are provided by the MPX Event Transmitter. To enable it, create a suitable `MPXEventTransmitter.xml` file and place it in the folder `<configuration repository path>/EventServices` for each StarTeam configuration that you want MPX-enabled.

The `MPXEventTransmitter.xml` file can be edited with a text editor when the StarTeam Server is not running or the server is not running with MPX enabled. When the server is running and MPX is already enabled, it can be modified dynamically via the StarTeam Server Administration utility. One MPX profile should be created for each deployed Message Broker. The profile designated as the “server default” is the one used by the event transmitter. Note

 **Note:** The Message Broker defined by the server default profile must be running when the StarTeam Server starts in order to run in MPX mode. See the *StarTeam Cross-Platform Client Help* for more information.

Next, enable the MPX File Transmitter for each configuration for which you want to use MPX caching functionality by creating a file called `MPXFileTransmitter.xml` in the same directory as the MPX Event Transmitter’s configuration file. The file transmitter’s configuration file does not typically require any customization.

There are no specific options in these configuration files to enable object caching. These files must simply be present when the StarTeam Server starts so that the appropriate transmitters are initialized, causing events to be broadcast through the MPX framework.

When you create a new StarTeam configuration via the StarTeam Server administration tool, the “template” files located in the `EventServices` directory are automatically copied to the appropriate `<configuration repository path>/EventServices` directory.

If you upgrade an existing StarTeam configuration to this release, your existing XML configuration files are automatically copied to the new location for the StarTeam release. However, be sure to start the StarTeam Server for a given configuration at least once before starting the root MPX Cache Agent for the same configuration. This is required to convert the old `CacheJournal.dat` file to a new format used for this release.

3 Configuring the Message Brokers

For each Message Broker, configure the corresponding `STMessageBroker68.ini` file. There are no specific options related to object caching for Message Brokers. The same configuration considerations apply as in previous releases. See the *StarTeam Cross-Platform Client Help* for more information.

4 Configuring the Root MPX Cache Agent

There are several configuration options for enabling object caching for root MPX Cache Agents. As before, a single root MPX Cache Agent process services one StarTeam configuration. If multiple MPX Cache Agent processes are configured to operate on the same machine, each requires its own configuration file, TCP/IP port, and local cache folder. If multiple MPX Cache Agent processes are to run as Windows services on the same machine, each service must be configured by running "`CacheAgentService.exe -register`" in a console window. See the *StarTeam Cross-Platform Client Help* for more information.

A root MPX Cache Agent is designated by creating an XML configuration file (`RootCAConfig.xml` in the figure above) that contains a `<RootCacheAgent>` group element. Without object caching, this group typically contains a single element called `<RootRepositoryPath>` that defines the root folder of the corresponding StarTeam configuration's "repository path". To support object caching, the root MPX Cache Agent requires two alternate elements that provide it with more information about the StarTeam configuration that it services.

```
<?xml version="1.0" ?>
<MPXCacheAgent version="2.0">
  <RootCacheAgent>
    <ServerConfigsFile>C:\Program Files\Micro Focus\StarTeam Server <version>
\starteam-server-configs.xml</ServerConfigsFile>
    <ConfigName>Sample Server Configuration</ConfigName>
    <Precharge>TipsOnly</Recharge>
  </RootCacheAgent>
  <Message broker>
    <Name>ActiveMQ MPX Transmitter</Name>
    <server_names>tcp:12.35.58.71:5101</server_names>
    <enable_control_msgs>echo</enable_control_msgs>
    <start_server_delay>10</start_server_delay>
    <socket_connect_timeout>10</socket_connect_timeout>
  </Message broker>
  <RequestPort>5201</Requestor>
  <MaxConnections>100</MaxConnections>
  <Cache types>
    <Object types>
      <ObjectType>Change</ObjectType>
      <ObjectType>Requirement</ObjectType>
      <ObjectType>Task</ObjectType>
      <ObjectType>$CustomComponents$</ObjectType>
    </Object types>
  </Cache types>
  <ListenAddresses>12.34.56.78, 21.43.65.87</Listen addresses>
  <InboundAddresses>12.34.56.78, 21.43.65.87</Inbound addresses>
  <MaxCatchupSize>100000000</MaxCatchupSize>
  <SharePolicy>Public</Share policy>
  <CachePath>C:\.MPXCacheAgent\Cache</Cyclepath>
  <MaxCacheSize>1000000000</MaxCacheSize>
  <MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
  <MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>
</MPXCacheAgent>
```

The `<ServerConfigsFile>` element specifies the full path name of the StarTeam Server configuration file. The `<ConfigName>` option defines the StarTeam configuration name that the root MPX Cache Agent will service. From these two parameters, the root MPX Cache Agent determines how to access the configuration's database, and it also determines the configuration's repository path.

At start-up, the root MPX Cache Agent will open the database and "pre-charge" its local cache with any missing object revisions of configured types. By default, the root MPX Cache Agent only pre-charges with

“tip” object revisions. You can specify that the pre-charge should include all object revisions by changing `<PreCharge>` to `All` instead of `TipsOnly`. (`TipsOnly` is the default value if `<PreCharge>` is not specified.) Keep in mind that pre-charging All object revisions requires significantly more local cache space. Alternatively, you can disable object revision pre-charging completely by setting `<PreCharge>` to `None`.

The `<ServerConfigsFile>` and `<ConfigName>` options are usually sufficient when the root MPX Cache Agent operates on the same machine as the StarTeam Server that it services. When the root MPX Cache Agent operates on a different machine, the following configuration points apply:

- The `<ServerConfigsFile>` option should use a UNC path that specifies the full path name of the server configuration file. Example:

```
<ServerConfigsFile>\\ProdServer\ST<version>\starteam-serverconfigs.xml</ServerConfigsFile>
```

The root MPX Cache Agent only reads the configuration file, therefore write access is not required.

- The machine on which root MPX Cache Agent operates must be able to access the configuration's database through the same database definition as the StarTeam Server. The root MPX Cache Agent fetches the database connection information from the StarTeam Server configuration file.
- You should define the `<RootRepositoryPath>` option to refer to the StarTeam configuration's repository path using a UNC path. Example:

```
<RootRepositoryPath>\\ProdServer\SampleServerConfiguration_SSE2012</RootRepositoryPath>
```

Finally, to enable object caching, the MPX Cache Agent supports a `<CacheTypes>` option group, which currently only allows a child element group called `<ObjectTypes>`. This option group is common to both root and remote MPX Cache Agents, hence it is specified as a child element of the outer-most `<MPXCacheAgent>` group. There are two basic ways to specify which object types to cache. The most common way to enumerate each object type within a `<ObjectType>` element. Example:

```
<?xml version="1.0" ?>
<MPXCacheAgent>
  ...
  <Cache types>
    <Object types>
      <ObjectType>Change</ObjectType>
      <ObjectType>Requirement</ObjectType>
      <ObjectType>Task</ObjectType>
      <ObjectType>$CustomComponents$</ObjectType>
    </Object types>
  </Cache types>
  ...
</MPXCacheAgent>
```

In this example, the MPX Cache Agent is configured to cache change request, requirement, and task objects. (Note that the “internal” object name is used – `Change` not `ChangeRequest`. These names correspond to `<type>.ssc` modules installed in the StarTeam Server's install directory.) Currently, there are six possible object types that can be cached: `Change`, `File`, `Folder`, `Requirement`, `Task`, and `Topic`. For each configured `<ObjectType>`, the root MPX Cache Agent will perform start-up pre-charging (as discussed above), catch-up and forwarding operations for upstream remote MPX Cache Agents, and object fetching for StarTeam clients.

Alternatively, you can specify that all cacheable object types are to be supported by setting the `AllTypes` attribute to `"True"` within the `<ObjectTypes>` group. Example:

```
<?xml version="1.0" ?>
<MPXCacheAgent>
  ...
  <Cache types>
    <ObjectTypes AllTypes="True"/>
  </Cache types>
```

```
...
</MPXCacheAgent>
```



Note: Some object types such as `File` and `Folder` do not possess many object properties, consequently caching these types may not be beneficial.

Cached objects are stored in the MPX Cache Agent's local cache as individual files. Because their contents are compressed, cached object files are typically small. Because of the large number of cached objects that are typically requested from a MPX Cache Agent at one time, it is not efficient to read each file directly from disk. Instead, MPX Cache Agent performance benefits greatly by caching object cache files in memory. Fortunately, due to their small size, a large number of cached objects will fit in memory at one time. Because of this performance benefit, memory caching is enabled by default.

Memory caching can also be explicitly controlled with two new configuration options shown below:

```
<?xml version="1.0" ?>
<MPXCacheAgent>
  ...
  <MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
  <MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>
  ...
</MPXCacheAgent>
```

The option `<MemoryCacheMaxSize>` controls the maximum amount of memory in bytes that the MPX Cache Agent uses for memory caching. When the MPX Cache Agent starts, the most-recently-used objects in its local cache are automatically loaded into memory. When the memory cache size is reached, new objects are added to the memory cache, but least-recently-used objects are removed from memory so that the memory cache size is not exceeded. If `<MemoryCacheMaxSize>` is not specified, it defaults to 100MB. Increasing this value results in the ability to cache more objects in memory at a cost of greater memory usage by the MPX Cache Agent process.

The option `<MemoryCacheMaxObjectSize>` defines the maximum size in bytes of an object that will be cached in memory. Regardless of whether they represent file contents or object properties, cache files larger than this size are not cached in memory. If `<MemoryCacheMaxObjectSize>` is not specified, it defaults to 10KB. Increasing this value allows larger objects to be cached in memory. However, fewer objects can be memory cached when the `<MemoryCacheMaxSize>` has been reached.

5 Configuring Remote MPX Cache Agents

To enable support object caching in a remote MPX Cache Agent, add the `<CacheTypes>` and `<ObjectTypes>` options defined above to the corresponding configuration file. The effect of the `<ObjectTypes>` option is slightly different for remote MPX Cache Agents as summarized below:

- When a remote MPX Cache Agent is configured with an `<ObjectTypes>` option, all upstream MPX Cache Agents (as defined by `<Content source>` groups) must minimally cache the same object types. When the remote MPX Cache Agent first starts, it “pings” each upstream MPX Cache Agent to determine the object types that it supports. If any upstream MPX Cache Agent is not configured to cache every `<ObjectType>` defined for the remote MPX Cache Agent, the remote MPX Cache Agent will report an error and terminate.
- If the configuration option `<ObjectTypes AllTypes="True"/>` is used, the remote MPX Cache Agent caches the “intersection” of object types being cached by all upstream MPX Cache Agents. If any upstream MPX Cache Agent is not enabled for object caching, or if there isn't at least one object type commonly cached by all upstream MPX Cache Agents, the remote MPX Cache Agent will report an error and terminate.

6 Enabling Object Caching for the StarTeam Cross-Platform Client

Several StarTeam clients support the use of MPX Cache Agents for file check-out operations, including the StarTeam Cross-Platform Client (CPC), command-line (stcmd), and bulk check-out utility (BCO). Any client that performs bulk “fetch item” or “refresh item” operations via the StarTeam SDK can use a MPX Cache Agent to fetch object properties.

To enable object caching for the StarTeam Cross-Platform Client:

1. Start the StarTeam Cross-Platform Client.
2. Click **Tools > Personal Options**. The **Personal Options** dialog box opens.
3. Select the **MPX** tab.
4. Check the **Enable MPX Cache Agent** check box.
5. You can specify a specific MPX Cache Agent to use or allow the client to locate the nearest MPX Cache Agent. Do one of the following:
 - Select the **Use MPX Cache Agent At** option button, providing both an address and port. The address can be the computer name or an IP address.
 - Select the **Automatically Locate the Closest MPX Cache Agent for Each MPX** option button and let the client do the work.
6. You can change the number of threads in the **Maximum Request Threads** field, but the default should be adequate for most users needs.
7. Under **Use MPX Cache Agent for**, ensure that the **Item properties** check box is selected in addition to the **File content** check box.
8. Click **OK**.



Note: When the **Item properties** check box is selected, the SDK automatically determines when object caching is beneficial and will use the MPX Cache Agent when performance is estimated to be faster than going to the StarTeam Server.

Configuring Clients

Any client can connect to ActiveMQ MPX Message Broker-enabled StarTeam Server, but not all of them can take advantage of MPX-features.



Note: Unless otherwise stated, references in this section to *client* refer only the StarTeam Cross-Platform Client.

Using ActiveMQ MPX from a Client

To configure support for ActiveMQ MPX on your workstation:

1. Start the client.
2. Click **Tools > Personal Options**. The **Personal Options** dialog box opens.
3. Select the **MPX** tab.
4. Select the **Enable MPX** check box to use ActiveMQ MPX with any MPX-enabled StarTeam Server connected to by the client.
5. Do one of the following:

Refresh manually (Shift+F5). Clear the **Automatic refresh with** check box.

Refresh automatically

1. Select the **Automatic refresh with** check box.
2. Set a minimum number of seconds between refreshes in the **Minimum delay of ___ seconds** field. The default is 5 seconds.
3. Set a maximum number of seconds between refreshes in the **Maximum delay of ___ seconds** field. The default is 30 seconds.

After every cache update, the application waits a minimum number of seconds before refreshing. This means that if cache updates are infrequent, the application performs a refresh almost immediately. However, if cache updates are frequent, the minimum refresh timer is constantly being reset and never reaches the number of seconds set for a refresh. In such cases, the next refresh occurs when the maximum number of seconds between refreshes forces a refresh.

6. Click **OK**. Your changes will take effect for all projects you open from this point on. Note that any projects that are currently open will be unaffected by your changes.

To stop using ActiveMQ MPX:

Clear the **Enable MPX** check box to stop support for any ActiveMQ MPX-enabled StarTeam Server connected to by the client.

Displaying MPX Status

When you open a project, the client's status bar displays the type of server configuration in use, the auto-refresh setting, and (for ActiveMQ MPX configurations) whether support for MPX is enabled.

The following shows the icons and wording that provide information about MPX when they appear on the status bar.

Yellow lightning bolt	Indicates that MPX is available and enabled for the currently selected project view. If Web Edition can use the default client profile for an MPX-enabled server and, therefore, take advantage of MPX, this icon appears in front of the server configuration's name in the browser window.
Gray lightning bolt	Indicates that MPX is available for the currently selected project view but that it has not been enabled in the client.
Red circle with a slash beside a yellow lightning bolt	Indicates that MPX is enabled for the currently selected project view, but something happened to break the connection. For example, the Message Broker may be stopped.
(no icon)	Indicates that MPX is not available for the currently selected project view.
Instant	Indicates that MPX's auto-refresh is turned on.
Auto	Indicates that your workstation's auto-refresh is turned on, but that MPX's auto-refresh is either turned off or unavailable. (Your workstation's auto-refresh option is on the Workspace tab of the Personal Options dialog box.)
Manual	Indicates that your workstation's auto-refresh is turned off and that MPX's auto-refresh is either turned off or unavailable. You must manually refresh the current project view by pressing F5.

Choosing a Non-default Connection Profile

In some cases, the client default profile for a given configuration may not be appropriate for every client. In those cases, the user can choose a profile other than the default.

To choose an ActiveMQ MPX connection profile other than the client default profile:

1. In the client, click **Project > Open command**. The **Open Project Wizard** opens.
2. Select the server configuration for which you wish to select a non-default client profile, and click **Server Properties**. The **Server Properties** opens and lists each profile defined in the Event Transmitter XML file for this server configuration.
3. Click **MPX Profiles**.
4. To examine the details of any profile, select the profile and click **Properties**.
5. Select the alternate profile that you wish to use and click **Set**. If you wish to restore the client default profile for this configuration, click **Restore Default** instead.
6. Click **Close** on this dialog box, and then click **OK** on the **Server Properties** dialog box.

When you open a project on a configuration for which you have chosen a non-default client profile, that profile will be used. Note that after a connection has been established, the client continues to use that messaging service even when it opens projects from other configurations.

Logging MPX Information in the Client Log

The StarTeam Cross-Platform Client can create a client log named `StarTeam.log`.

To record ActiveMQ MPX information in this log:

1. Start the client.

2. Click **Tools > Personal Options**. The **Personal Options** dialog box opens.
3. Select the **Workspace** tab.
4. Select the **Log MPX Events** check box.
5. Click **OK**.

To review the StarTeam log, click **Tools > StarTeam Log**.

Using MPX Cache Agent from the StarTeam Cross-Platform Client and IDEs

If ActiveMQ MPX and a MPX Cache Agent have been installed and configured, you can use the MPX Cache Agent from your StarTeam Cross-Platform Client or an IDE based on StarTeam Cross-Platform Client or .NET components. Using a “network-near” MPX Cache Agent should provide faster check-out operations.

Enabling MPX Cache Agent Use

The StarTeam Server to which you connect must be MPX-enabled. However, the StarTeam Cross-Platform Client does not have to enable ActiveMQ MPX to take advantage of MPX Cache Agent. ActiveMQ MPX provides properties about files and other items. MPX Cache Agent provides file and/or object caching. Using both would be typical but is not mandatory.

MPX Cache Agent use can be enabled either through personal options or in server properties, specific to the server you are editing. Personal options would only be referenced if no specific settings on the server have been made.



Note: If ActiveMQ MPX is not enabled, the MPX Cache Agent cannot be auto-located. It must be configured with a specific address and port.

Enable the MPX Cache Agent via Server Properties

Use the following procedure to enable the MPX Cache Agent through the server properties:

1. Start the StarTeam Cross-Platform Client.
2. Click **Project > Open**. The **Open Project Wizard** dialog box opens.
3. Select the server configuration for which you wish to enable MPX Cache Agent use, and click **Server Properties**. The **Server Properties** dialog box opens.
4. Select the **MPX Cache Agent** tab.
5. Under **Use MPX Cache Agent for**, select whether the MPX Cache Agent will be used for file caching (**File Content**)/object caching (**Item Properties**). When the **Item properties** check box is selected, the SDK automatically determines when object caching is beneficial and will use the MPX Cache Agent when performance is estimated to be faster than going to the StarTeam Server.
6. You can specify a specific cache agent to use or allow the client to locate the nearest cache agent. Do one of the following:
 - Select the **Use MPX Cache Agent At** option button, providing both an address and port. The address can be the computer name or an IP address.
 - Select the **Automatically Locate the Closest MPX Cache Agent for Each MPX** option button and let the client do the work.
7. You can change the number of threads in the **Maximum Request Threads** text box, but the default should be adequate for most users needs.

8. Click **OK**.

Enable the MPX Cache Agent via Personal Options

Use the following procedure to enable the MPX Cache Agent through personal options:

1. Start the StarTeam Cross-Platform Client.

2. Click **Tools > Personal Options**.

The **Personal Options** dialog box opens.

3. Select the **MPX** tab.

4. Select the **Enable MPX Cache Agent** check box.

5. Select **Automatic refresh with** to enable the automatic refresh of the application window by way of MPX. The default minimum is 30 seconds, and the default maximum is 0– seconds. If this option is unchecked, you must refresh manually (Shift+F5).

6. Under **Use MPX Cache Agent for**, select whether the MPX Cache Agent will be used for file caching (**File Content**)/object caching (**Item Properties**). When the **Item properties** check box is selected, the SDK automatically determines when object caching is beneficial and will use the MPX Cache Agent when performance is estimated to be faster than going to the StarTeam Server.

7. You can specify a specific cache agent to use or allow the client to locate the nearest cache agent. Do one of the following:

- Select the **Use MPX Cache Agent At** option button, providing both an address and port. The address can be the computer name or an IP address.
- Select the **Automatically Locate the Closest MPX Cache Agent for Each MPX** option button and let the client do the work.

8. You can change the number of threads in the **Maximum Request Threads** text box, but the default should be adequate for most users needs.

9. Click **OK**.

Checking out Files with the MPX Cache Agent

The visible advantage to using MPX Cache Agent is the improved speed of file check-out operations. The more files you check out, the more advantage you will gain from MPX Cache Agent. Over time, more and more of the files will come from MPX Cache Agent, reducing the strain on StarTeam Server. As a result, the check-out speed should continue to improve until all files are available from MPX Cache Agent.

For a particular check-out operation, you can see how many files are being sent by StarTeam Server directly and how many are being sent by MPX Cache Agent by displaying the check-out statistics.

To monitor check-out statistics using MPX Cache Agent:

1. Select the files to be checked out.

2. Click **File > Check Out**. The **Check Out** dialog box opens.

3. Select the **Show Checkout Statistics** check box.

4. Choose any other option settings that are appropriate to your check-out operation.

5. Click **OK**.

6. During the check out process, you will see a dialog which indicates the file names and the location from which they are coming (StarTeam Server or MPX Cache Agent). After the operation completes, the **Checkout Statistics** dialog box provides a summary.



Note: From some IDEs, you can also **Show Checkout Statistics**.

Using MPX Cache Agent with Bulk Checkout Utility

You can use MPX Cache Agent with the Bulk Checkout (bco) utility. See the *StarTeam Cross-Platform Client Help* for more information.

Running MPX Components

Most of the ActiveMQ MPX start-up and shut down routines are performed automatically by the operating system and StarTeam Server. This chapter describes how to manually start and stop the Message Broker on Microsoft Windows and Linux platforms. It also explains how to start and stop MPX Cache Agent on those platforms.

Running Message Broker on Microsoft Windows

On Microsoft Windows platforms, the Message Broker is installed as a Microsoft Windows service. When this service is installed, the installer asks whether you wish to create an automatic or a manual service.

- If you choose automatic, the corresponding service will be started by Microsoft Windows each time the system is initialized.
- If you choose manual, you must manually start the service each time the system is initialized.

Both automatic and manual services are automatically stopped when the system is shutdown. Both automatic and manual services can be manually started and stopped as needed. The procedures for starting and stopping the Message Broker are provided in the following topics.

Starting a Message Broker

1. On the computer where the Message Broker is installed, click **Start > Settings > Control Panel > Administrative Tools > Services**. On Microsoft Windows XP systems, click **Start > Control Panel > Performance and Maintenance > Administrative Tools > Services**.
2. Select the service named `Micro Focus ActiveMQ Message Broker`.
3. Click **Start**.

When the Message Broker starts, it reads the `ActiveMQMessageBroker.ini` configuration file. On Windows systems, this file is typically located in the `C:\Program Files\Micro Focus\Message Broker` folder. Options in this file tell the Message Broker what TCP/IP port (end point) to accept connections on, and which other Message Brokers (if any) to establish communication with to form a Message Broker cloud. See *Configuring a Message Broker Cloud* for details on the contents of the `ActiveMQMessageBroker.ini` file.

Stopping a Message Broker

Under most conditions, a Message Broker runs continuously. However, it may be necessary to stop a Message Broker. For example, you may choose to move a Message Broker to a different computer or remove a Message Broker from a Message Broker cloud. If the Message Broker that you stop is serving clients, those clients continue to access the server configurations, but without using ActiveMQ MPX.

1. You should first notify users that ActiveMQ MPX will be unavailable.
2. On the computer where the Message Broker is installed, click **Start > Settings > Control Panel > Administrative Tools > Services**. On Microsoft Windows XP systems, click **Start > Control Panel > Performance and Maintenance > Administrative Tools > Services**.
3. Select the service named `Micro Focus ActiveMQ Message Broker`.

4. Click **Stop**.

Running the ActiveMQ MPX Message Broker on Linux

ActiveMQ is the file with the start/termination scripts for ActiveMQ. On Linux, you need to manually start or stop the Message Broker. To do this:

1. Navigate to the folder containing the start/termination scripts. Usually it is under <your ActiveMQ installation Dir>/bin .
2. Run the appropriate script with the parameter `start` to start the corresponding daemon, or `stop` to stop an existing daemon. For example:

```
./activemq start
./activemq stop
./activemq status
```

3. To run the broker in the console

```
activemq console
```

Running MPX Cache Agents

To run the MPX Cache Agents successfully, start the applications you have installed in the following order:

1. Message Brokers
2. StarTeam Server (the server configuration starts the transmitters)
3. Root MPX Cache Agent
4. Remote MPX Cache Agent

For more details about starting MPX components see, *Dependencies - Startup Order for MPX Components*.

Running MPX Cache Agent On Microsoft Windows

MPX Cache Agents can be run as service or console applications. Up to 25 Root and Remote MPX Cache Agents can run as services on the same computer. However, only the first one installed on a computer will be registered as a service. It is registered as manual service with the display name `MPX Cache Agent` and an internal name of `CacheAgentService`. You will need to register the others.

MPX Cache Agent is stopped automatically when the computer is shut down.

Running MPX Cache Agent as a Service

If a MPX Cache Agent is running as a service, you can start and stop it manually using the **Control Panel Services** utility. Click **Start > Settings > Control Panel > Administrative Tools > Services** to see the list of services.

Using the same utility, you can change the MPX Cache Agent service to run automatically when you start Microsoft Windows.

You can also control the MPX Cache Agent service from the command line by running the `CacheAgentService.exe`. Use any of the following syntaxes:

```
CacheAgentService -start [ configFile ] [ -log logFile ] [ -verbose ]
```

```
CacheAgentService -register [ Manual | Auto ] [ configFile ] [ -dependson list ] [ -log logFile ] [ -name serviceName ] [ -verbose ]
```

```
CacheAgentService -unregister
```


configFile	The default configuration file is <code>CacheAgentConfig.xml</code> . If you use multiple configuration files, each one must specify unique values for <code>CachePath</code> and <code>RequestPort</code> so that MPX Cache Agent services do not interfere with each other's operation.
-dependson list	Specifies service dependencies for the new MPX Cache Agent service. The list must be a quoted, space-separated list of internal (not display) names of the services on which the MPX Cache Agent service will depend. (A service's internal name is its registry key within the Microsoft Windows registry.) The most common dependency for a MPX Cache Agent is to make it dependent on the Message Broker service running on the same machine. The Message Broker service's display name is typically <code>ActiveMQ Message Broker for CM Hub</code> , but its internal name is typically <code>ActiveMQMessageBroker</code> . Consequently, to make a MPX Cache Agent depend on the Message Broker service on the same computer you would use: <pre>-dependson ActiveMQMessageBroker</pre>
-log logFile	Specifies a log file name other than the default, which is <code>CacheAgentService.log</code> .
-name serviceName	Specifies the display name for the service in the Control Panel's Services utility. The default is <code>MPX Cache Agent</code> .
-register	Register the service with the specified start mode (<code>Manual</code> or <code>Auto</code>), optionally with a specific configuration file at startup.
-start	Starts the service, optionally with a specific configuration file.
-unregister	Removes a service. For example, if you change the MPX Cache Agent from <code>Manual</code> to <code>Auto</code> , you would unregister it and re-register it. A service must be stopped before it can be unregistered.
-verbose	Causes another more detailed log to be generated. It defaults to <code>CacheAgentService-debug.log</code> .

Examples

Below is an example command-line to register an auto-start MPX Cache Agent service with the name "Prod1 Root CA", dependent on the ActiveMQ MPX Message Broker, with the default log file name, and verbose logging enabled:

```
CacheAgentService -register Auto c:\CAConfigs\Prod1RootCA.xml -dependson "ActiveMQMessageBroker" -name "Prod1 Root CA" -verbose
```

As with the register command, the default service name is "MPX Cache Agent". That means that the name parameter must be used when you unregister a service that has a non-default name. For example, to un-register the service used in the last example:

```
CacheAgentService -unregister -name "Prod1 Root CA"
```

Log File

When MPX Cache Agent runs, it creates a log file in its installation folder. The default file name for the log is `CacheAgentService.locale.log` where `locale` is something like `en-US`.

If a file with that name already exists, it is renamed to include a time stamp:

```
CacheAgentService_YYYYMMDDhhmmss.location.log
```

Its log can be viewed from a browser by entering a URL using the following syntax:

```
http://host:port/log
```

host Identifies the computer on which the MPX Cache Agent is running.

port Provides its configured port number. The default port is 5202.

Running MPX Cache Agent As a Console Application

`CacheAgentApp.exe` can be used instead of the service application. Use it when multiple MPX Cache Agents operate on the same machine against different StarTeam Server configurations.

For this scenario, each MPX Cache Agent uses a different request port and different local cache paths.

When running the `CacheAgentApp.exe`, you can use the following syntax:

```
CacheAgentApp [ -c configFile ] [ -l logfile ] [ -v off | on ]
```

Parameter	Description
<code>-c configFile</code>	Starts the MPX Cache Agent as a console application, optionally with a specific configuration file. The default configuration file is <code>RootCAConfig.xml</code> or <code>RemoteCAConfig.xml</code> , depending on the type of the MPX Cache Agent.
<code>-l logfile</code>	Specifies a log file name other than the default, which is <code>CacheAgentApp.log</code> .
<code>-v</code>	Add more detail to the log. The settings are <code>off</code> or <code>on</code> . <code>Off</code> is the default.

Log File

When MPX Cache Agent runs, it creates a log file in its installation folder. The default file name for the log is `CacheAgentService.locale.log` where `locale` is something like `en-US`.

If a file with that name already exists, it is renamed to include a time stamp:

```
CacheAgentService_YYYYMMDDhhmmss.location.log
```

Its log can be viewed from a browser by entering a URL using the following syntax:

```
http://host:port/log
```

host Identifies the computer on which the MPX Cache Agent is running.

port Provides its configured port number. The default port is 5202.

Running MPX Cache Agent on Linux

To run a MPX Cache Agent on a Linux system, use the following `start` command:

```
cacheagentapp -c RootCacheAgentConfig.xml -d start
```

To stop a MPX Cache Agent on a Linux system, use the following `stop` command:

```
cacheagentapp -c RootCacheAgentConfig.xml -d stop
```

Server Log Entries

Each time you start a server configuration, it creates a new server log file to record all activity. If a server log file already exists, the existing file is renamed before the new file is created.

The server log file has been renamed to reflect your language environment. For example, a `Server.en-US.log` file indicates the US English language. Note that even if your language is not US English, a `Server.en-US.log` file will always be created for use by Micro Focus support.

This section shows sample ActiveMQ MPX-related messages that may appear in a server log file. These messages are typically prefixed with `ActiveMQ MPX`.



Note: Each message written to the server log file includes a line number, and a date-time stamp. This information has been omitted from the following sample for the purposes of clarity.

Start-Up Messages

When you start a server configuration, the system records all start-up messages in the server log file. When you start an ActiveMQ MPX configuration, the server log file also records start-up information for the Event Transmitter. The following sample shows the typical entries that are made in the server log file when you start an ActiveMQ MPX-enabled server configuration. (The ActiveMQ MPX-specific messages appear in boldface type.)

```
Found Event Transmitter configuration of type "ActiveMQ MPX Transmitter" with
ServerDefault settings:
```

```
server_write_timeout=30
server_names=tcp:localhost:61616
socket_connect_timeout=10
server_start_delay=10
transport.commandTracingEnabled=true
enable_control_msgs=echo
server_max_reconnect_delay=10
project=Starbase
server_start_max_tries=1
server_read_timeout=30
server_keep_alive_timeout=30
```

```
Connecting to broker(s): tcp://localhost:61616,
EventTransmitter: Initialized PubSub Library
EventTransmitter: Created connection to StarTeam Message Broker
EventTransmitter: Created encryption cipher
MPX File Transmitter: Using hive index file: c:\repository\testActiveMQ
\HiveIndex\hive-index.xml
MPX File Transmitter: Using journal file: c:\repository\testActiveMQ
\CacheJournal.dat
MPX File Transmitter: Hive #10 mapped to folder: c:\repository\testActiveMQ
\DefaultHive\Archives\
MPX File Transmitter: Using data transfer rate of 256 kbps
MPX File Transmitter: Using packet transmission delay of 32 ms
```

Reconnect Messages

If an MPX-enabled server configuration loses the connection to its Message Broker, it records that event in the server log file. A server configuration will always attempt to re-establish communication with the Message Broker and it will wait for the Message Broker to come back online.. The number of retries made

and the time between retries depends on the parameters set in the Event Transmitter XML file's server default profile.

The following sample shows the typical entries that are made in the server log file when an MPX configuration loses and then reconnects to its Message Broker.

```
49 00000001 2015-06-30 16:07:05 ActiveMQ MPX: Lost connection to broker
50 00000001 2015-06-30 16:07:32 ActiveMQ MPX: Resumed connection to broker
```

Troubleshooting ActiveMQ MPX

To determine whether your Microsoft Windows-based ActiveMQ MPX system is operating correctly, you can perform the following steps.

1. Review the following configuration files to ensure that the server addresses and endpoints are correct:

MPXEventTransmitter.xml	For the Event Transmitter installed for each StarTeam Server.
FileTransmitter.xml	For the File Transmitter installed for each StarTeam Server.
activemq.xml	For each Message Broker.
ActiveMQMessageBroker.ini	For each Message Broker.
RootCAConfig.xml	For each Root MPX Cache Agent.
RemoteCAConfig.xml	For each Remote MPX Cache Agent.

2. If they are not already running, start the Message Brokers.
3. For each Message Broker you start, start a ActiveMQ MPX-enabled server configuration that communicates with that Message Broker.
4. For each server configuration you start, review its server log file. If the Event Transmitter has any problems connecting to the Message Broker, the error messages will be written to the server log file (for example, `Server.en-US.log`), which is located in the root folder of the server configuration's repository.
5. Start a client and ensure that support for ActiveMQ MPX is enabled for your workstation.
6. Enable client ActiveMQ MPX options:
 - a) In your client, click **Tools > Personal Options**. The Personal Options dialog box opens.
 - b) Select appropriate options on the **Workspace** and **MPX** tabs. The StarTeam Cross-Platform Client has MPX options on the **Workspace** tab. This option allows the StarTeam log file to include MPX information. The log file can be viewed at any time by clicking **Tools > StarTeam Log**. It also has settings for enabling and disabling MPX.

Only the StarTeam Cross-Platform Client and IDEs based on StarTeam Cross-Platform Client and .NET components have options for MPX Cache Agent.
 - c) Click **OK**. Test these settings.
7. In your client, open a view from an ActiveMQ MPX-enabled server configuration. If MPX is enabled in both the client and the server configuration, a yellow lightning bolt appears in the status bar.
8. Ensure that the machines running the different message brokers have their clocks synchronized. Failure to do this could cause message expiration and clients may not receive messages.

Diagnosing a Message Broker

ActiveMQ MPX Message Brokers generate a log file named `Activemq.log`, located in `data` folder under the `Message Broker` installation folder. For example, at `C:\Program Files\Micro Focus\ActiveMQ Message Broker\data`.

Index

A

access rights 14

C

cache agent
 configuring a remote 41
choosing a non-default connection profile 59
component descriptions 13
configure clients 58
configuring components 15

D

data encryption 14
displaying MPX status 58
documentation 5

E

enabling MPX on multiple StarTeam Server configurations 30
event transmitter
 about 31
 startup 31
 XML file format 32

F

file transmitter
 about 36
 startup 37
 XML file format 37
framework and architecture 8

I

installation overview 15

L

logging MPX information in the client log 59

M

Message Broker
 changing the endpoint 26
 clouds 21
 communication 22
 configuring 25
 configuring a cloud 26
 configuring two in a fail-over configuration 27
 controlling connections 28
 diagnosing 69
 managing 21

message routing in clouds 22
planning 21
routing unconnected clouds 23
running on Linux 64
running on Microsoft Windows 63
starting on Microsoft Windows 63
stopping on Microsoft Windows 63
tracing 28
using with a firewall 25
volume considerations 24

MPX

components 10
 configuration
 ActiveMQ MPX 18
 ActiveMQ MPX and StarTeamMPX together 19
 ActiveMQ MPX only 20

MPX Cache Agent

use with bulk checkout utility 62
check out files with 61
configuring a root 40
enable via personal options 61
enable via server properties 60
enabling 60
managing 38
operations 39
planning 38
remote XML parameters 46
reviewing status and log information 49
root or remote XML parameters 42
root XML parameters 48
running 64
running as a console application 66
running as a service 64
running on Linux 66
running on Microsoft Windows 64
using from StarTeam Cross-Platform Client 60
using with clients 49
XML parameters 42

O

object caching
 about 50
 configuration 52
 how it works 50
 required components 52
overview 7

P

preface 5
product support 6
profiles with multiple connections 35

R

running MPX components 63

S

security 14

server log entries

about 67

reconnect 67

start-up 67

startup order for MPX components 17

SupportLine 6

T

transmitters

connection profiles 31

enabling for server configurations 29

managing 29

XML files 29

troubleshooting 69

U

user authentication 14

using MPX from a client 58