StarTeamMPX Administrator's Guide

Borland® StarTeam® 13.0

Borland® MICRO FOCUS®

Micro Focus
575 Anton Blvd., Suite 510
Costa Mesa, CA 92626

Copyright © 2012 Micro Focus IP Development Limited. All Rights Reserved. StarTeam™ contains derivative works of Borland Software Corporation, Copyright © 1995-2009 Borland Software Corporation (a Micro Focus company).

MICRO FOCUS, the Micro Focus logo, and Micro Focus product names 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.

BORLAND, the Borland logo, StarTeam™, and Borland product names are trademarks or registered trademarks of Borland Software Corporation 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.

ST13.0-MPX

August 2012

PDF

Contents

Chapter 1
Preface **5**
Contacting Micro Focus Support 5
Documentation Conventions 6

Chapter 2
Overview **7**
The StarTeamMPX Framework and Architecture . . . 9
Component Descriptions 11
StarTeamMPX Security 13
Data Encryption 13
User Authentication and Access Rights 14
Message Broker Profiles 14
Overview of the StarTeamMPX Installation 15
Overview of Configuring StarTeamMPX Components 15
Dependencies - Startup Order for MPX Components. 17

Chapter 3
Managing Message Brokers **19**
Planning for Message Brokers 19
Understanding Clouds 20
How Communication Is Established Between Message Brokers 21
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 26
Configuring a Message Broker Cloud 26
Changing the Endpoint of a Message Broker . . . 27
Configuring Two Message Brokers in a Fail-Over Configuration 29
Enabling Tracing for Message Brokers 30
Controlling Connections 30

Chapter 4
Managing the Transmitters **31**
Configuration-specific Transmitter XML Files 31
Enabling Transmitters for Server Configurations . 32
Enabling MPX on Multiple StarTeam Server Configurations 33
Understanding Connection Profiles 34
Understanding the Event Transmitter 35
Startup 35
The XML File's General Format 36
Using Profiles with Multiple Connections 41
Understanding the File Transmitter 42
Startup 42
The XML File 43

Chapter 5
Managing Cache Agents **45**
Planning for the Cache Agents 46
Cache Agent Operations 48
Configuring a Root Cache Agent 49
Configuring a Remote Cache Agent 50
Parameters 51
Reviewing Status and Log Information 58
Using Cache Agent with the Clients 59
Object Caching 59
How Object Caching Works 60
Components Needed for Object Caching 63
Configuring Object Caching 64
Configuring the MPX Transmitters 64
Configuring the Message Brokers. 65
Configuring the Root Cache Agent 65
Configuring Remote Cache Agents 69
Enabling Object Caching for the Cross-Platform Client. 69

Chapter 6
Configuring Clients **71**
Using MPX from a Client 71
Displaying MPX Status 72
Choosing a Non-default Connection Profile 73
Logging MPX Information in the Client Log 74
Working with the StarTeamMPXCommands.txt File . 74
Using Cache Agent from the Cross-Platform Client and IDEs 75
Enabling Cache Agent Use. 75
Server Properties 75
Personal Options 76
Checking out Files with Cache Agent 77
Using Cache Agent with bco. 77

Chapter 7
Running MPX Components **79**
Running Message Broker on Windows 79
Starting a Message Broker 80
Stopping a Message Broker 80
Running a Message Broker on Linux 81
Running Cache Agents 81
Running Cache Agent On Windows 82
Running Cache Agent as a Service 82
Examples 83
Log File 83
Running Cache Agent As a Console Application. 85
Log File 85
Running Cache Agent on Linux 85

Appendix A
Server Log Entries **87**
Start-Up Messages 88
Reconnect Messages 89

Appendix B
Troubleshooting StarTeamMPX **91**
Diagnosing a Message Broker. 92

Index **93**

Chapter 1

Preface

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

Important The online manuals are distributed in Adobe's Portable Document Format (.pdf), which requires the free Acrobat Reader to display them. The reader is available from Adobe's Web site at (www.adobe.com).

The documentation for the StarTeam SDK includes a StarTeamMPX Programmer's Guide. The SDK also includes the EventMonitor.jpx sample project (for Borland JBuilder™) which has some MPX examples.

Support

Borland (a Micro Focus company) is committed to providing world-class services in the area of consulting and technical support. We have over 30 years of experience in supporting developers and enterprise customers. Our 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 support worldwide, delivering timely, reliable service to ensure every

customer's business success. For more information about Micro Focus' support services, go to http://supportline.microfocus.com.

From the Web site, you can also access many newsgroups where users exchange information, tips, and techniques. This Micro Focus Community, available at http://community.microfocus.com, provides access to product specific information, articles, code examples, and news.

When contacting support, be prepared to provide complete information about your environment, the version of the product you are using, and a detailed description of the problem.

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

Documentation Conventions

The documentation uses the following conventions.

Select File > Exit	Indicates a menu selection followed by a submenu selection. The greater-than symbol (**>**) separates the commands to be selected from subsequent menus. In this case, select File from the menu bar, then select Exit from the drop-down menu.
Fixed-Space Font	Text appearing in Courier font represents information that you need to type and messages from the system.
italics	Information that you replace with the names of your files, child folders, etc.
Bold	Information that you must use exactly as shown (if you use it).
[ ]	Square brackets surround optional syntax.
|	A vertical bar separates mutually exclusive choices.

Note Identifies supplemental information.

Tip Identifies information on alternative procedures or other helpful but nonessential information.

Important Identifies information that is essential to the completion of a task.

Caution Identifies actions that may result in loss of data or procedures that must be followed to ensure that data is *not* lost.

Chapter

2

Overview

StarTeamMPX 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 StarTeamMPX components:

- Message Broker
- Event Transmitter

StarTeam Enterprise Advantage licenses support all of the StarTeamMPX components:

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

StarTeamMPX 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 StarTeamMPX features.

Changes to the server configuration's repository are broadcast in an encrypted format to StarTeam clients and Cache Agents through a publish/subscribe channel. The Event Transmitter broadcasts encrypted messages about changes to objects, such as change requests, and the 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 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 Cache Agents are StarTeamMPX clients that rely on messages and files transmitted by the File Transmitter, they also serve other StarTeamMPX clients as they check out files.

The StarTeamMPX technology offers a number of key benefits, including:

- **Bandwidth multiplication:** 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.

  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.

  For the person who builds software products, check-out operations are a very high percentage of the client-to-server traffic. Using the Bulk CheckOut (bco) command-line utility with a Cache Agent can speed up the time it takes to create a build.

- **Performance acceleration:** Because the event caching modules reside on the same computer as the client and the 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.

- **Burst control:** 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.

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

StarTeam client:	Can take advantage of:
StarTeam Cross-Platform client	Event and file/object caching
Bulk CheckOut (bco) command-line utility	File caching
IDEs based on Cross-Platform and .NET components (such as the Eclipse integration and the Visual Studio .NET integration)	Event and file/object caching
Web Edition	Event caching (if the default client profile is for a Message Broker that is accessible from the computer running the IIS)

The StarTeamMPX Framework and Architecture

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

- **Messaging engine component:** The 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 Event Transmitter and 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 Message Broker. StarTeam clients can receive and cache event messages.

Figure 2.1 presents an overview of the StarTeamMPX 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 Message Brokers serving many clients.

Figure 2.1 StarTeamMPX System Architecture for Event Transmission

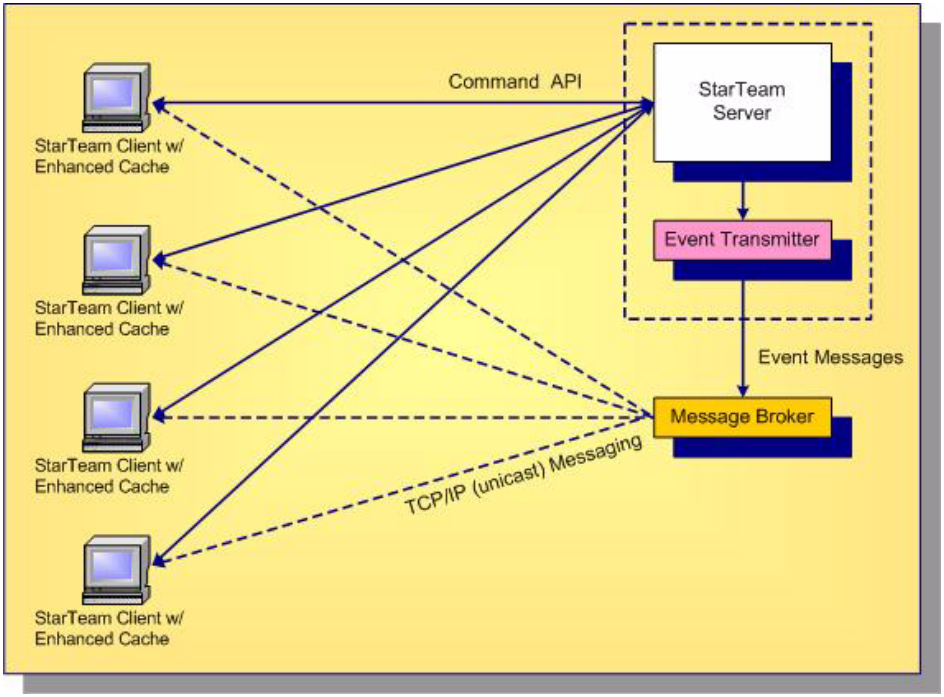
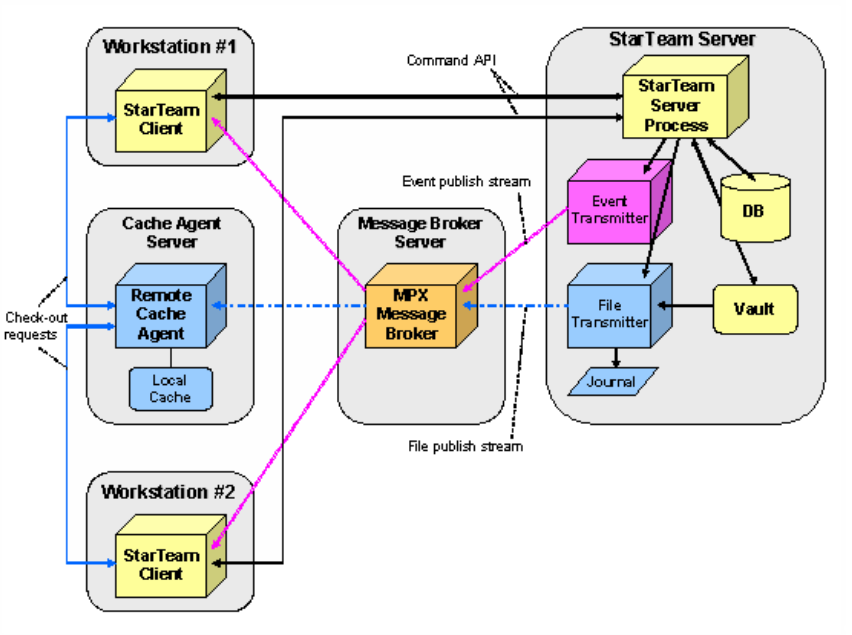


Figure 2.2 and Figure 2.3 present an overview of the StarTeamMPX system architecture for file transmission. In Figure 2.2, a single Cache Agent is operating on its own server, servicing check-out requests for two Cache Agent-aware clients. Depending upon the individual needs of a particular site or facility, there may be several MPX-enabled server configurations, Message Brokers, Root Cache Agents, and Remote Cache Agents serving many clients.

Figure 2.2 StarTeamMPX System Architecture for File Transmission-Single



Cache Agent

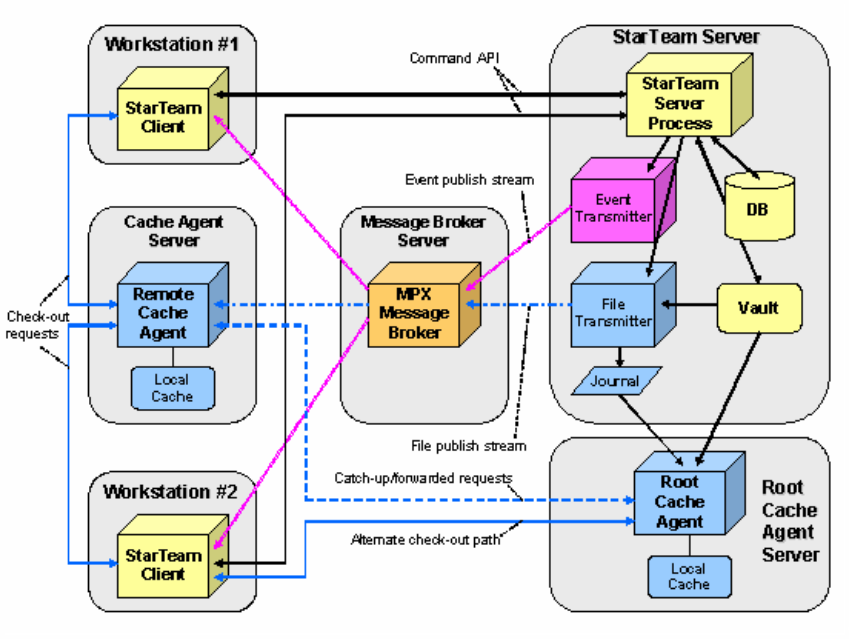
You can organize Cache Agents hierarchically or “tiered” to support distributed teams and improve performance over slow or unreliable network connections. It also allows Cache Agents to forward request “misses” and to “catch-up” with content that was missed during network or process outages. An example of tiered Cache Agents is depicted in Figure 2.3.

The tiered caching capability requires the operation of a specially-configured Cache Agent known as the Root Cache Agent. The Root 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 Cache Agent also requires access to the journal file (CacheJournal.dat) maintained by the File Transmitter. The journal file provides the information needed by the Root Cache Agent to access individual file revisions.

In Figure 2.3, one Remote Cache Agent is operating on its own computer and is being used by two clients. While all file revision transmissions are broadcast

by the File Transmitter, the fact that the Remote Cache Agent is “tiered” to the Root Cache Agent allows cache misses and catch-up requests to be forwarded to the Root Cache Agent from the Remote Cache Agent.

Figure 2.3 StarTeamMPX System Architecture for File Transmission- Root Cache Agent



A 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 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 Help* guide or the Cross-Platform Client online help for more information about this utility which improves check-out speeds both with and without Cache Agent’s help.

Component Descriptions

This section provides a short description for each StarTeamMPX 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 Event Transmitter and the File Transmitter. It notifies the Event Transmitter each time a subscribed event occurs, and sends it relevant details about the event.

Event Transmitter

The Event Transmitter broadcasts events of interest to clients. The 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 Event Transmitter is installed when you install StarTeam Server.

File Transmitter

The File Transmitter broadcasts file contents and object properties to one or more Remote Cache Agents by means of a Message Broker. Like the Event Transmitter, the 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 Event Transmitter or encrypted content messages from the 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.

Cache Agents

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

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

A Root Cache Agent handles requests forwarded from Remote 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 Cache Agent.

By using “network-near” 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 or .NET components, or the Bulk CheckOut (bco) command-line utility to retrieve files from a Cache Agent. The Cache Agent's file/object caching is independent of client processes because the Cache Agent operates as a separate process. Consequently, a client can fetch files that were broadcast while it was not operational.

StarTeamMPX Security

The StarTeamMPX 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 Cache Agents through a Message Broker

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

All data sent by the transmitters or Cache Agents is encrypted. Each Event Transmitter has its own encryption key. When the server configuration starts an Event Transmitter, it creates a unique encryption key for that instance of

the Event Transmitter. When a client opens a project, the server configuration sends the client the 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 File Transmitter are encrypted. The content is stored in encrypted format by 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 Cache Agent, granting someone the "see item and its properties" right implicitly virtually grants them a "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 Cache Agent check-out. For environments in which this difference in security "interpretation" matters, you should not deploy Cache Agent or deny the "see item and its properties" right for users who should not check-out the corresponding files.

Message Broker Profiles

MPX profiles that specify appropriate Message Brokers are specified in the Event Transmitter’s configuration file and used by the server configuration and clients. It is best to use a network-near broker.

The StarTeam administrator must create MPX profiles to fit the needs of each general location in distributed groups. For example, all the clients using the same Remote Cache Agent may be network-near the same Message Broker and, therefore, use that Message Broker’s profile.

For more information, see “Using Profiles with Multiple Connections” on page 41.

Overview of the StarTeamMPX Installation

When installing StarTeamMPX 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 StarTeamMPX components:

1. Install StarTeam Server. The Event and File Transmitters are installed automatically with StarTeam Server.
2. Install the StarTeam Message Broker. You can run multiple instances of the Message Broker.
3. Install the Root 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 Cache Agent. This is the same installer as the Root Cache Agent. You can run one or more copies on remote machines.

For complete details about installing StarTeamMPX components, refer to the *StarTeam Installation Guide* (ST_Install_en.pdf).

Overview of Configuring StarTeamMPX Components

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 StarTeamMPX.

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

- Review the Root Cache Agent archive path: If using a Root Cache Agent on a different machine than the StarTeam Server, you should update the "Root Cache Agent archive path" for each hive to reflect the path with which the Root 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 StarTeam 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 *server_installation_path*/EventServices/*server_configuration*/. Normally these files are copied automatically from the template files

(MPXEventTransmitterTemplate.xml and MPXFileTransmitterTemplate.xml), which are installed at *server_installation_path*/EventServices/.

You typically do not need to edit the MPXFileTransmitter.xml file. Its presence in the *server_installation_path*/EventServices/*server_configuration*/ 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 StarTeam 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 StarTeam Server Administration tool only after the server configuration is running and the StarTeamMPX 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.

- StarTeam Message Broker: Review the file STMessageBroker68.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 Cache Agent: You must typically edit the configuration file. Use a text editor such as Notepad for editing this file.

- 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 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 Message Broker that the Root Cache Agent will use. You may also need to adjust the <CachePath> property.

  If running a Root 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 Cache Agent service for each configuration.

- Remote Cache Agent: You must edit the configuration file for each Remote 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.
- Modify the Cross-Platform client to use StarTeamMPX using the Tools > Properties command. For more information, see Chapter 6, “Configuring Clients,” on page 71.
- Root and Remote 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 Cache Agent is installed as an auto-start service and uses a Message Broker on the same computer, the Root Cache Agent may start more quickly than the Message Broker. In this case, the Root Cache Agent fails. The fix is to make the Root 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 Cache Agent to create the CacheJournal.dat file. The first time you start a StarTeam configuration after a 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 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 Cache Agent fails to start. Once the CacheJournal.dat file exists, the root 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 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 Cache Agent fails to start. However, if one of the Root Cache Agents 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 Cache Agents or clients can access it.
- The Cache Agents must be running before the clients can retrieve files from them.

Chapter 3

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* (ST_Install_en.pdf).

Planning for Message Brokers

Planning considerations for the Message Broker include:

- At least one Message Broker is required for StarTeamMPX 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 Chapter 4, “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”.

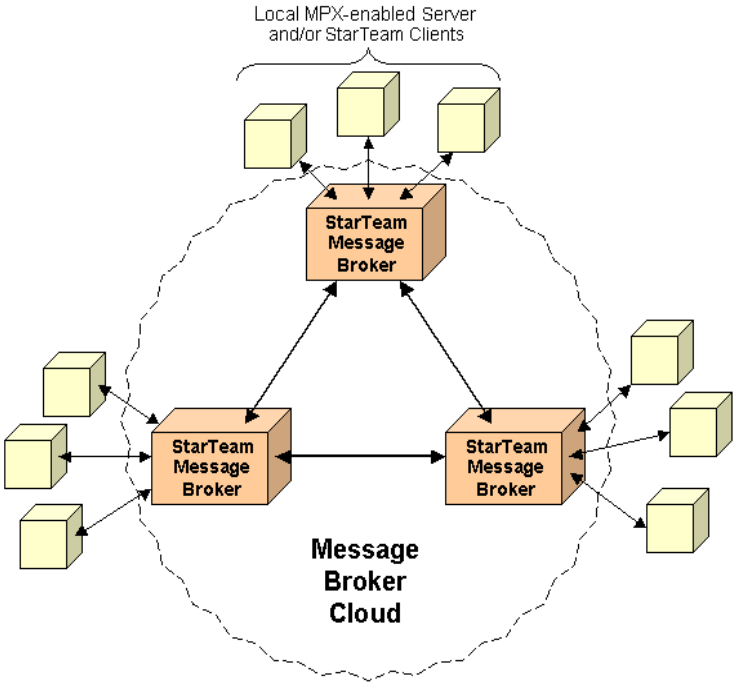
The cloud allows a subscriber to use StarTeamMPX features with server configurations that reside in another geographic location.

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

shows an example of a Message Broker cloud. Some components in the StarTeamMPX architecture are omitted for simplicity.

Example Message Broker Cloud



How Communication Is Established Between Message Brokers

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 STMessageBroker68.ini configuration file located in the same folder as the executable STMessageBroker68.exe. The optional server_names parameter in the STMessageBroker68.ini file identifies other Message Brokers in the Message Broker cloud. When a Message Broker first starts, it reads the STMessageBroker68.ini file and attempts to connect with each of the Message Brokers listed in server_names. 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.

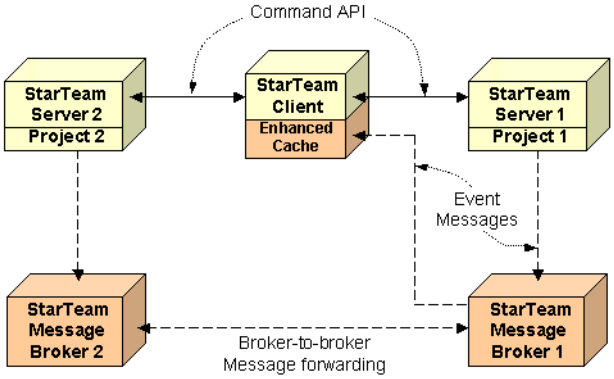
Message Routing in Message Broker Clouds

When a subscriber first connects to an MPX-enabled server configuration, it uses a connection profile (described in “Using Profiles with Multiple Connections” on page 41) 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 Figure 3.1, 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 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.

Figure 3.1 Client with projects open on two MPX-enabled Servers



Routing in Unconnected Message Broker Clouds

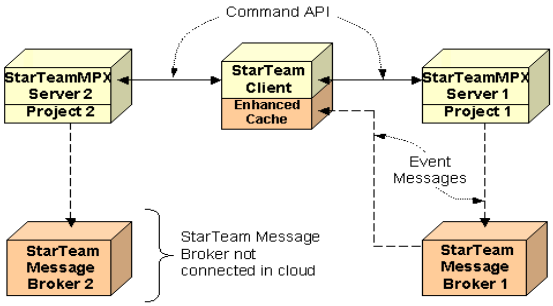
In a properly configured Message Broker cloud, messages from all server configurations to which the subscriber is connected are routed to the appropriate Message Broker for that subscriber. However, in unconnected clouds, not all Message Brokers are aware of other Message Brokers and cannot forward messages 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 3.2). In this case, if the client chooses an MPX profile the specifies Message Broker 1 when it opens Project 1, the client receives messages for only the server configuration in the first cloud. The client treats the server configuration in the second cloud as if it did not support StarTeamMPX. The client performs all functions properly for such “disjoint” server configurations, but it does not receive the performance and instant notification benefits provided by StarTeamMPX.

Figure 3.2 Message routing in an unconnected Message Broker cloud



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 StarTeamMPX architecture, each subscriber using a unicast connection profile creates a TCP/IP connection to a Message Broker. The Event Transmitter and 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). The configurations settings needed for selecting this operation are discussed in“Configuring Two Message Brokers in a Fail-Over Configuration” on page 29.
- **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 StarTeamMPX functionality. Issues for operating Message Brokers external to a firewall are discussed in the following section.

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 StarTeamMPX 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 STMessageBroker68.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 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 a STMessageBroker*xx*.ini file that contains startup parameters. This file must be located in the same folder as the Message Broker executable STMessageBroker*xx*.exe.

This folder is typically "C:\Program Files\Borland\Message Broker <version>" on Windows platforms.

On Linux platforms, it is "/opt/MessageBroker*xx*".

If the STMessageBroker*xx*.ini file is missing, the Message Broker uses predefined default values for all of the parameters.

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* (ST_Install_en.pdf) for instructions.
2. For each Message Broker you want to include in the cloud, open the associated STMessageBroker68.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:

setopt 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.

setopt server_names tcp:ProdServer1:5110, tcp:ProdServer2:5120

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 STMessageBroker68.ini file.

To change the endpoint number of a Message Broker:

1 Make the change to the STMessageBroker68.ini file for that Message Broker:

a Open the STMessageBroker68.ini file in a text editor.

b Add or edit the following line:

setopt 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 “_node” to designate the primary IP address of the local host independent of its host name; you can use the keyword "_any" to have the Message Broker listen on all available IP addresses of the local host

- endpoint is the new endpoint you want the Message Broker to use; the default is 5101

  For example:

  setopt conn_names tcp:_node:5523

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 STMessageBroker68.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 utility to edit the affected profiles:

  a From Server Administration, select the server configuration.

  b Select Tools > Administration > Configure Server.

  c From the Event Handlers tab, select StarTeamMPX Transmitter as the event handler.

  d From the Profile list box, select an affected profile.

  e Click Modify to open the Event Handlers Profile Properties dialog.

  f In the Profile Properties list box, double-click server_names.

  g In the Event Handler Property dialog, edit the appropriate endpoints in the Property Value text box.

  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 the server is not currently running or not running in MPX mode):

  a Open the MPXEventTransmitter.xml file in a text editor.

  b Edit the server_names parameter in each affected unicast profile to reflect the new endpoint.

  c Save and close the file.

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

4 For each Cache Agent that accesses this Message Broker, edit the MessageBroker group to change the appropriate server_names parameter.

a Open the 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:

setopt server_names tcp:*other_server*:5101

to the STMessageBroker68.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 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 StarTeamMPX 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 out of memory), you can turn on tracing by adding the following lines to its STMessageBroker68.ini file:

setopt Trace_Level Debug

setopt Trace_File *log_file_path*

setopt Trace_Flags prefix,timestamp

The case of the option names is not important.

The *log_file_path* can be any file path, for example, C:\temp\MBTrace.log.

The "Debug" trace level is the highest (most verbose), so the trace file will grow quickly. The levels "info" and "warning" are successively lower trace levels (less verbose).

Controlling Connections

You can control the total number of connections between Message Brokers and the total number of client connections to a Message Broker.

The Max_Server_Conns option controls the total number of connections that a Message Broker can have with other Message Brokers. This is all Message Broker-to-Message Broker connections, whether initiated inbound or outbound. If unspecified, the value is -1, which means unlimited. Since each Message Broker has its own .ini file, each Message Broker could have a different setting for this option. Consequently, it does not constrain the overall "cloud size". For example, even if every Message Broker's value was set to 2, you could "daisy chain" a list of Message Brokers together. This option only applies to Message Brokers, so it can be used only in STMessageBroker68.ini files.

By default, it is set to 10:

```
setopt Max_Server_Conns    10
```

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 STMessageBroker68.ini.

By default, it is set to 2000:

```
Max_Client_Conns    2000
```

Chapter

4

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 chapter 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 StarTeamMPX installation, template files named MPXEventTransmitterTemplate.xml and MPXFileTransmitterTemplate.xml are installed on the same computer as StarTeam Server. Their default

locations depend on the operating system type, as shown in the following table.

Operating System	Default location for MPXEventTransmitterTemplate.xml
Microsoft Windows Server 2003 R2 SP2, (32-bit)	C:\Program Files\Borland\StarTeam Server <version>\EventServices\
Microsoft Windows Server 2008 (32- and 64-bit)	
Microsoft Windows Server 2008 R2 (64-bit)	
Red Hat Enterprise Linux 5 Server, (32-bit)	/opt/starteamserver<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 MPXEventTransmitterTemplate.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 EventServices folder.

The configuration-specific XML files are stored in subfolders of the 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" on page 27 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 Cache Agent.

Enabling Transmitters for Server Configurations

During the StarTeam Server installation, default template files (MPXEventTransmitterTemplate.xml and MPXFileTransmitterTemplate.xml) are installed in the *server_installation_path*/Event Services folder. The server installation also creates folders for the MPXEventTransmitter.xml and

MPXFileTransmitter.xml files of existing 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 *server_installation_path*/Event Services/*server_configuration* folder. However, you must have an Enterprise Advantage server in order to use the 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 Event Services/*server_configuration* 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 Event Services/*server_configuration* folder for the new server configuration, copy the template files from *server_installation_path*/Event Services to the new folder, rename the template files to MPXEventTransmitter.xml and MPXFileTransmitter.xml, and edit them.

The Event Transmitter enables basic MPX messaging. You can edit the default MPXEventTransmitterTemplate.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 File Transmitter requires the Event Transmitter and controls the transmission of files and objects. You typically do not need to edit the File Transmitter's MPXFileTransmitter.xml file. Having a copy of it in the Event Services/*server_configuration* folder is enough to enable the File Transmitter and, in turn, the Cache Agents.

Enabling MPX on Multiple StarTeam Server Configurations

When the Cache Agent is installed, you are asked for the location of the server configuration's repository. The default is the StarDraw repository, if StarDraw is installed on the server. But you can change that when you install the Cache Agent, and make another server configuration the "default". This server configuration will use the default Request Port of 5201. There is no option to change this during the Cache Agent install.

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 Cache Agent configuration file that points to the StarTeam configuration's repository and uses a unique Request Port. You then need to

create a 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 Cache Agent’s install directory and make a copy of RootCAConfig.xml. Call it MyCAConfig.xml.
4. Get 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. Change RequestPort to something other than 5201 or a port already in use. Save the changed file.
6. At the DOS prompt go to the 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 Cache Agent service for the server configuration.
7. Start the Cache Agent service.

Here is an example CacheAgentService invocation for the “My” server configuration:

.\CacheAgentService –register Manual “C:\Program Files\Borland\StarTeamMPX Cache Agent <version>\MyCAConfig.xml” –name “StarTeam My Cache Agent” –log “MyCALog.log”-verbose

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 next few sections cover the Event Transmitter and its XML file.

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" on page 87 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.

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. It 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.

The XML File's General Format

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

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<StExternHandlerModule version="1.0">
        <Handler>
                <Name>...</Name>
                <Description>...</Description>
                <Properties>
                        <ClientDefault>...</ClientDefault>
                        <ServerDefault>...</ServerDefault>
                </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.

Table 4.1 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 4.1 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 utility to set a profile as the server default.

Table 4.1 Event Transmitter Parameters

Parameter	Description
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 utility 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. **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 ServerDefault or ClientDefault parameter. **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 "A unicast/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. **Properties** This is a group inside the Profile group. Its inner parameters define the connection details of the profile. The parameters within this group specify the connection details of the profile. The most common options that can be used within the Properties group and their usage are described below. 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 Borland developer support services (http://supportline.microfocus.com). The parameters in the Properties group are listed in Table 4.2.

Table 4.2 Parameters for the Properties Group Inside the Profile Group

Parameter	Description
server_names	Designates the protocol, address, and port number of one or more Message Brokers. The general syntax for this parameter is: *tcp*:*servername*:*endpoint* where servername is the TCP/IP address or computer name of the computer running a Message Broker. endpoint is the port number on which the Message Broker accepts connections. The default endpoint for a Message Broker connection is 5101. Examples: For a Message Broker running on a computer with the TCP/IP address of 12.34.56.78 and port number 5320, the server_names value would be: <server_names>tcp:12.34.56.78:5320</server_names> 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: <server_names>tcp:HostA:5101,tcp:HostB:5101 </server_names> The Event Transmitter and any clients using a connection profile with this server_names value will first attempt to connect to the Message Broker on the computer named HostA, using port 5101. If a connection to that Message Broker is unsuccessful, the process will attempt to connect to the Message Broker on HostB using port number 5101. _**random** The _random option is used with <server_names>. 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 _random, followed by a comma, before the list of pooled Message Brokers. For example: _random, tcp:*HostA*:5101, tcp:*HostB*:5101, tcp:*HostC*:5101 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.
server_names (continued)	Consequently, as StarTeam clients connect and execute the same <server_names> value using the _random option, their Message Broker connections will be distributed among the available Message Brokers, causing automatic load balancing to take place. Note that when the _random option is used, if a selected Message Broker is unavailable, another Message Broker is randomly selected, until an available Message Broker is found.
project	Specifies a publish/subscribe “universe” name. The default value is Starbase and **should not be changed**.

Table 4.2 Parameters for the Properties Group Inside the Profile Group

Parameter	Description
enable_control_msgs	The default is echo. Specifies the types of control messages that the StarTeamMPX process (and application clients) will honor. Only the echo control message should normally be enabled, so this value defaults to echo and normally should not be modified.
server_keep_alive_timeout	integer; number of seconds. The default is 30. 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. Minimum value is zero, which disables the feature.
server_read_timeout	integer; number of seconds. The default is 30. 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. Minimum value is zero, which disables the feature. **Note:** The client_read_timeout option in the STMessageBroker68.ini file should be set higher than the server_read_timeout option in all profiles in the MPXEventTransmitter.xml file. The client_read_timeout option by default is 45 seconds. This causes the clients to drive keep-alive logic, freeing-up processing from the Message Brokers.
server_write_timeout	integer; number of seconds. The default is 30. 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. Minimum value is zero, which disables the feature.
server_start_max_tries	Integer. The default is 1. Specifies how many times the Event Transmitter or application client will traverse the server_names list during a connect sequence, before giving up and deciding that no messaging service is available. Minimum value is zero, which disables the feature.
server_start_delay	Integer; number of seconds. The default is 10. Specifies how long (in seconds) to wait between traversals of the server_names 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. Minimum value is zero, which disables the feature.

Here is an example unicast profile:

```
<Profile>
   <Name>Unicast Off-site</Name>
   <Description>The MPX Message Broker profile for clients using
     unicast.
      </Description>
   <Properties>
      <server_write_timeout>30</server_write_timeout>
      <server_names>tcp:123.45.6.78:5101</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_read_timeout>30</server_read_timeout>
      <server_keep_alive_timeout>30</server_keep_alive_timeout>
   </Properties>
</Profile>
```

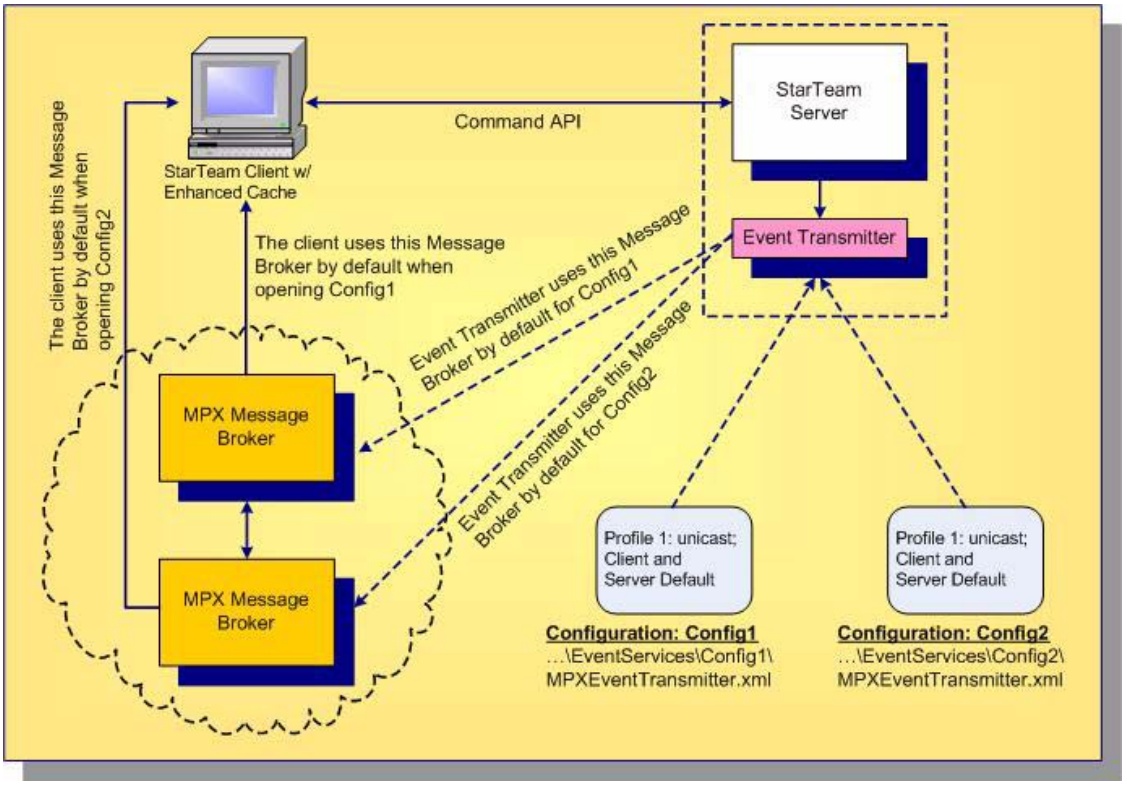
If you manually edit an Event Transmitter configuration-specific XML file, the changes will take effect after you save the updated file and restart the corresponding configuration. If you use the Server Administration tool, changes go into effect immediately for all new client connections.

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 utility.

Using Profiles with Multiple Connections

In Figure 4.1, 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 MB, one as the client and server default, and the other as an optional profile.

Figure 4.1 Example Configuration-specific Transmitter XML Files



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 that the client can also define custom connection properties that are stored locally on the client workstation. Client configuration options are described in Chapter 6, “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 Cache Agent uses this information and broadcasts files directly from the Native-II Vault.

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.

The XML File

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

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<StExternHandlerModule version="1.0">
        <Handler>
      <Name>StarTeamMPX 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 optional properties in Table 4.3 should rarely need to be overridden.

Table 4.3 File Transmitter Parameters

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 CacheJournal.dat 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 CacheAgent 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.

Chapter 5

Managing Cache Agents

MPX Cache Agents provide persistent caching of StarTeam information in geographically distributed locations. By retrieving information from a network-near 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 Cache Agents also reduces demand on the StarTeam server, increasing its scalability and responsiveness for other requests.

MPX Cache Agents can cache *file contents*, thereby improving the performance of *check-out* commands. Cache Agents also support *object caching*, which allows StarTeam clients to fetch properties for change requests, tasks, and other objects from a specially-configured Cache Agent. As with file content caching, using a network-near 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 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 Cache Agent provides several important services:

- It supports the same fetch requests that all 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 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 Cache Agent also acts as an “upstream” Cache Agent, providing “downstream” Remote Cache Agents with a place to forward cache misses. This increases the “hit rate” of Remote Cache Agents and improves their effectiveness.
- The Root Cache Agent provides a “catch up” API, allowing Remote Cache Agents to proactively fetch files that they missed while they were not running. Because the Journal is time-sequenced, a Remote 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 Cache Agents to be effective even over unreliable network connections.

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

A Remote 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 Cache Agent or another “upstream” Cache Agent.
- Use a different upstream Cache Agent for the file and object contents of each server configuration it is tracking.
- Automatically find the Root Cache Agent through MPX poll messages. This allows a Remote 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 Cache Agent automatically finds the Root 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 Cache Agent can be used immediately, since cache misses will be forwarded on demand.

Planning for the Cache Agents

Issues to be considered as you do Cache Agent planning include:

- On Windows, a Cache Agent can run as a service or as a console application. More than one Cache Agent can be run as a service on each computer if appropriate.
- A Root 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 Cache Agent must be connected to a Message Broker.

- The Root 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 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 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 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 Cache Agent requires access to a Message Broker.
- Remote Cache Agents should be installed in each geographic location that can benefit from improved file check-out performance. One approach is to install a Cache Agent in each network environment in which local users can access it over a high-speed LAN. (Example: Install two Remote Cache Agents at headquarters, one each for the engineering and quality assurance teams, a third Remote Cache Agent at the Chicago office, and a fourth at the London office.)
- Installing a Remote 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 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 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 Cache Agent’s cache is configurable.
- The cache for a Cache Agent does not have to be backed up and can be deleted, if necessary, when the Remote Cache Agent is not running.

- Clients can be configured to use a specific Cache Agent by specifying that Cache Agent’s host name (or IP address) and port number. Alternatively, some clients can be configured to locate an appropriate Cache Agent automatically. If multiple Cache Agents are available, the client automatically chooses the “network-nearest” Cache Agent. This feature keeps administrative overhead to a minimum and allows the automatic detection of new Cache Agents by clients.
- You can use a single Remote Cache Agent without deploying a Root 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 Cache Agent there is no file-catch-up or request forwarding.)

Cache Agent Operations

The following sections provide information for understanding how Cache Agent operates with the Message Broker, other 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 Cache Agent starts up for the first time, it reads the CacheJournal.dat file from beginning to end. This makes the Cache Agent aware of the most recently added and modified files and objects and, thereby, able to return them in fetch requests. The Root 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 Cache Agent starts, depending on its settings, it can update its cache with data from its logical "parent" Cache Agent, usually the Root 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 Cache Agent requests a file or object that the Remote 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 Cache Agent.
- If the Remote 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 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 Cache Agents provide project filtering capabilities. Specifically, you can specify a list of "Project" parameters in the "ContentSource" section of the Remote 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 Cache Agent

By default, the name of the Root Cache Agent configuration file is RootCAConfig.xml. If you run only one instance of 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 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 Cache Agent per computer. See “Running Cache Agents” on page 81 for more details.

An example of the configuration file to establish a Root Cache Agent is shown below. The root element for the Cache Agent configuration file must be MPXCacheAgent. The XML elements are summarized in Table 5.1 and Table 5.3.

```
<?xml version="1.0" ?>
<MPXCacheAgent>
    <RootCacheAgent>
        <ServerConfigsFile>C:\Program Files\Borland\StarTeam Server
<version>\starteam-server-configs.xml</ServerConfigsFile>
        <ConfigName>StarDraw</ConfigName>
        <Precharge>TipsOnly</Precharge>
    </RootCacheAgent>

     <MessageBroker>
     <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>
    </MessageBroker>

    <RequestPort>5201</RequestPort>
    <MaxConnections>100</MaxConnections>

    <CacheTypes>
```

```
    <ObjectTypes>
      <ObjectType>Change</ObjectType>
      <ObjectType>Requirement</ObjectType>
      <ObjectType>Task</ObjectType>
      <ObjectType>$CustomComponents$</ObjectType>
    </ObjectTypes>
   </CacheTypes>
   <ListenAddresses>12.34.56.78, 21.43.65.87</ListenAddresses>
   <InboundAddresses>12.34.56.78, 21.43.65.87</InboundAddresses>
   <MaxCatchupSize>100000000</MaxCatchupSize>
   <SharePolicy>Public</SharePolicy>
  <CachePath>C:\.MPXCacheAgent\Cache</CachePath>
  <MaxCacheSize>1000000000</MaxCacheSize>
   <MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
   <MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>
</MPXCacheAgent>
```

Configuring a Remote Cache Agent

A Remote 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 Cache Agent is shown below. The absence of a RootCacheAgent element denotes the configuration for a Remote Cache Agent.

```
<?xml version="1.0" ?>
<MPXCacheAgent>
  <MessageBroker>
    <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>
  </MessageBroker>

  <RequestPort>5201</RequestPort>
  <MaxConnections>100</MaxConnections>
   <CacheTypes>
    <ObjectTypes>
      <ObjectType>Change</ObjectType>
      <ObjectType>Requirement</ObjectType>
      <ObjectType>Task</ObjectType>
```

```
            <ObjectType>$CustomComponents$</ObjectType>
        </ObjectTypes>
    </CacheTypes>

        <ListenAddresses>12.34.56.78, 21.43.65.87</ListenAddresses>
        <InboundAddresses>12.34.56.78, 21.43.65.87</InboundAddresses>
        <MaxCatchupSize>100000000</MaxCatchupSize>
        <SharePolicy>Public</SharePolicy>
    <CachePath>C:\.MPXCacheAgent\Cache</CachePath>
    <MaxCacheSize>1000000000</MaxCacheSize>
        <MemoryCacheMaxSize>100000000</MemoryCacheMaxSize>
        <MemoryCacheMaxObjectSize>10000</MemoryCacheMaxObjectSize>

    <ContentSource>
        <ServerGUID>be5ee3b0-c719-49c6-a1a1-f493764a03f5</
ServerGUID>
        <UpstreamCache>
            <UpstreamHost>ProdServer1</UpstreamHost>
            <UpstreamPort>1123</UpstreamPort>
        </UpstreamCache>
    </ContentSource>

    <ContentSource>
        <ServerGUID>79408139-1768-4031-9ddd-7f1b095c94e7</
ServerGUID>
            <Projects>
                <Project>FelixTools</Project>
                <Project>Bank*</Project>
                <Project>Insurance*West*</Project>
            </Projects>
        <UpstreamCache>
            <AutoLocate/>
        </UpstreamCache>
    </ContentSource>
</MPXCacheAgent>
```

Parameters

The XML elements for both Root and Remote Cache Agents are summarized in Table 5.1, Table 5.2, and Table 5.3.

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.

Table 5.1 Parameters Used by Any Cache Agent

Parameter	Description
CacheCheckInterval	Integer; number of seconds. The default is 60. The frequency with which the 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 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 Cache Agent either by “downstream” 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 Cache Agent uses the StarTeam Server configuration’s cache.
CacheTypes	Specifies the type of information that will be cached by the Cache Agent. If this group is not specified, the Cache Agent caches file contents only. **ObjectTypes** Specifies one or more object types to be cached by the Cache Agent. If this group is specified with the attribute **AllTypes="True"**, then all object types are cached by the 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 Cache Agent. The object types listed in the Remote Cache Agent must be the same or a subset of the object types listed in the Root 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 $CustomComponents$ 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 Cache Agent.

Table 5.1 Parameters Used by Any Cache Agent

Parameter	Description
InboundAddresses	Comma-separated list of IP addresses. By default, the Cache Agent uses a list of its machine's IP addresses. Returned by the Cache Agent in "poll" responses, this list also appears as the "Inbound Addresses" value on the Cache Agent's status page. In cases where some addresses should be hidden from clients (for example, they are not routable) or the 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 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 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 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 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 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 2^{63}-1. For a Root Cache Agent, this parameter constrains the maximum number of files returned in a catch-up request. For a Remote Cache Agent, this parameter constraints 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.

Table 5.1 Parameters Used by Any Cache Agent

Parameter	Description
MaxConnections	Integer; number of connections. The default is 100. The range is 1 to 1000. The maximum number of simultaneous connections that the Cache Agent will accept. This value controls the maximum number of request handler threads used by the 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 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 Cache Agent's MessageBroker group should contain a value for the server_names parameter. A Remote 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 is a “public” Cache Agent that will respond to poll messages. (See SharePolicy.) **server_names** Defines the publish/subscribe messaging service to be used by the 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 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 (5101 by default). Example: tcp:MBServer1:5101. **enable_control_msgs** Specifies the types of control messages that the StarTeamMPX process (and application clients) will honor. Only the echo control message should normally be enabled, so this value defaults to echo and normally should not be modified. **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, 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 Cache Agent’s status page.

Table 5.1 Parameters Used by Any Cache Agent

Parameter	Description
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 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 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 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 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 Cache Agents that are caching data for a specific StarTeam server GUID. Public Cache Agents will respond to such requests when they are caching the requested server's content, but private Cache Agents will not.

Table 5.2 Parameters Used by Remote Cache Agent Only

Parameter	Description
CatchupCheckInterval	Integer; number of seconds. The default is 300. The interval in which a remote 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.

Table 5.2 Parameters Used by Remote Cache Agent Only

Parameter	Description
ContentSource	Each ContentSource group specifies a StarTeam server configuration that the Remote Cache Agent will monitor for new file content via MPX messages. Optionally, this group specifies an "upstream" Cache Agent that will be contacted for catch-up and forwarding requests. A Remote 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 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 Cache Agent will not receive new file content. To locate the GUID for a specific server configuration: 1 From the Server Administration dialog, select the server. 2 Click the Properties icon on the toolbar or select Server > Server Properties from the menu. These actions display the Properties dialog. 3 Copy the GUID from the server configuration’s Properties dialog 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 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 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 Cache Agents tracking project P1 store the new file revision. A Remote Cache Agent tracking only project P2 will not receive the broadcast. However, "pull through" caching, explained more below, allows the Remote Cache Agent that is tracking only project P2 to obtain the file revision anyway. When a Remote 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 Cache Agent forwards the request to the appropriate Root 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 Cache Agents by "push" caching, but it does not limit the files a Remote Cache Agent can store via "pull through" (request forwarding) caching.

Table 5.2 Parameters Used by Remote Cache Agent Only

Parameter	Description
	UpstreamCache This group specifies that an upstream Cache Agent will be contacted for catch-up and forwarding requests for content related to the corresponding StarTeam server. The upstream 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 Cache Agent, while AutoLocate requests polling. **AutoLocate** Indicates that the Root Cache Agent for the corresponding StarTeam server configuration is to be automatically located and used as the upstream Cache Agent.Remote Cache Agent to continue periodic polls for a Root Cache Agent until one is found. It is an empty tag and mutually exclusive with UpstreamHost. **UpstreamHost** Explicitly identifies the host name or IP address of the upstream Cache Agent to be used for content related to the corresponding StarTeam server. This parameter is mutually exclusive with AutoLocate. **UpstreamPort** Integer. The default is 5201. The range is from 1 to 65535. Only meaningful if UpstreamHost is specified, this parameter specifies the port number of the upstream Cache Agent.
PrechargeSize	Integer. The default is MaxCatchupSize. Specifies the maximum size of a Remote Cache Agent's first catch-up operation, which is used to precharge its cache. It defaults to the Cache Agent's MaxCatchupSize, but it can be specified larger or smaller than that value. If this value is zero, the Remote Cache Agent does not perform an initial precharge operation.

Table 5.3 Parameters Used by Root Cache Agent Only

Parameter	Description
RootCacheAgent	Group for Root 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 Cache Agent will cache. It must be used in conjunction with the <ServerConfigsFile> option. **PreCharge** Specifies whether the root Cache Agent should "pre-charge" its local cache with missing object revisions by scanning the database at start-up time. If <PreCharge> is "None", a pre-charge is not performed. If <PreCharge> is "TipsOnly" (the default), only tip revisions are added to the local cache. If <PreCharge> is "All", all object revisions are added to the local cache. This option is ignored if the <ObjectTypes> option is not specified. **RootRepositoryPath** Required. The full path name of the StarTeam server configuration's repository folder. This option is needed only when ServerConfigsFile is not used. **RootHiveIndexPath** The full path of the StarTeam Server's Native-II hive index file. The default is *RootRepositoryPath*/HiveIndex/hive-index.xml. **RootJournalPath** The full path name of the StarTeam server configuration's cache journal file, which is created by the File Transmitter. The default is *RootRepositoryPath*/CacheJournal.dat.

Reviewing Status and Log Information

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

To check the status of an active Cache Agent:

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

   http://*host***:***port*/**status**

   host is the 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 Cache Agent's log file from a browser.

If verbose mode was used when the Cache Agent was started (see "Running Cache Agent as a Service" on page 82 and "Running Cache Agent As a Console Application" on page 84 for more details), you can use the following syntax:

http://*host***:***port***/debuglog**

to display a more detailed log.

Using Cache Agent with the Clients

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

See "Using Cache Agent from the Cross-Platform Client and IDEs" on page 75 for information about using Cache Agent when checking out files with this client.

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

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 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 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:

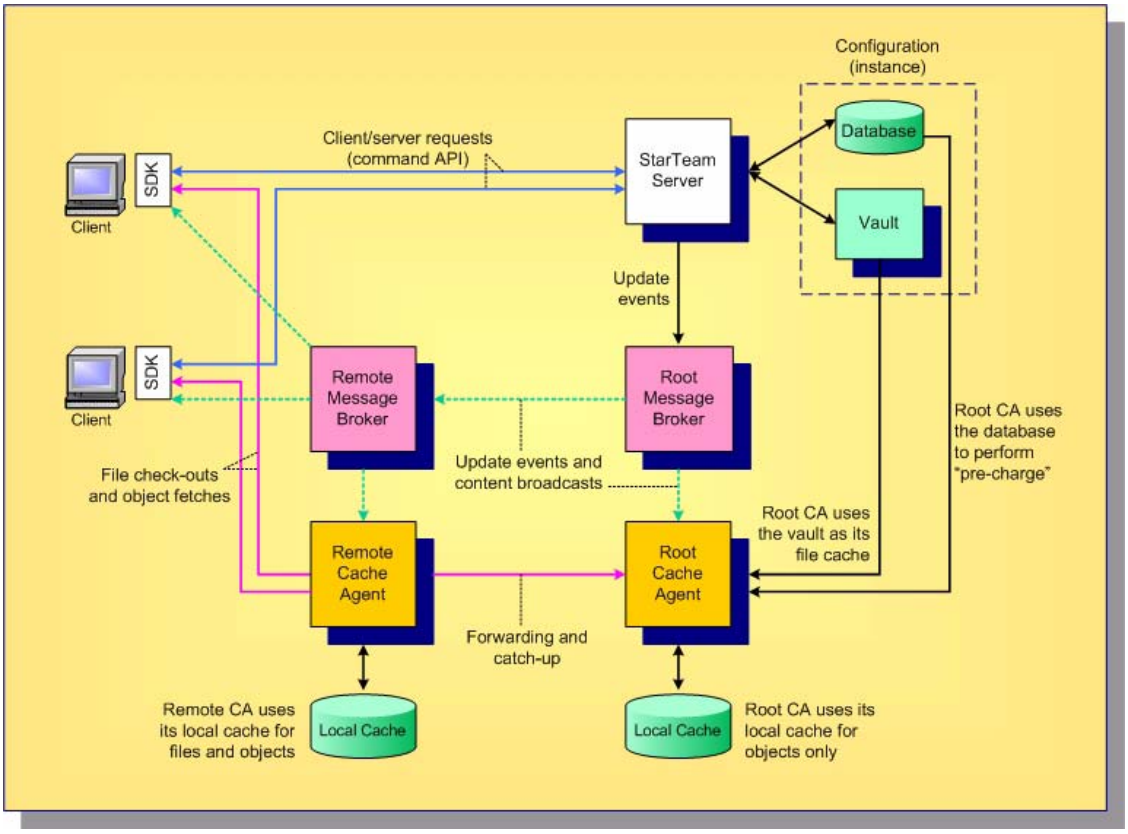
> *Addressed By, Addressed In, Addressed In View, Attachment Count, Attachment IDs, CR Number, Category, Closed On, Component, Created By, Created Time, Deleted By, Deleted Time, Description, Dot Notation, End Modified Time, Entered By, Entered On, External Reference, Fix, Last Build Tested, Modified 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 StarTeam MPX components. The diagram below illustrates a typical deployment with all major MPX components.

Figure 5.1 Typical MPX Deployment



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 Cache Agent** is deployed network-near to the StarTeam Server, and a **remote 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 Cache Agents. (A root 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 Cache Agent usage, it can check-out files from a network-near 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 Cache Agent and remote Cache Agents. Clients that enable Cache Agent access can subsequently fetch specific object revisions from a network-near Cache Agent, again at potentially greater speed. The StarTeam SDK utilizes an adaptive performance monitoring algorithm to determine when Cache Agent access is beneficial. Consequently, it only uses a 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

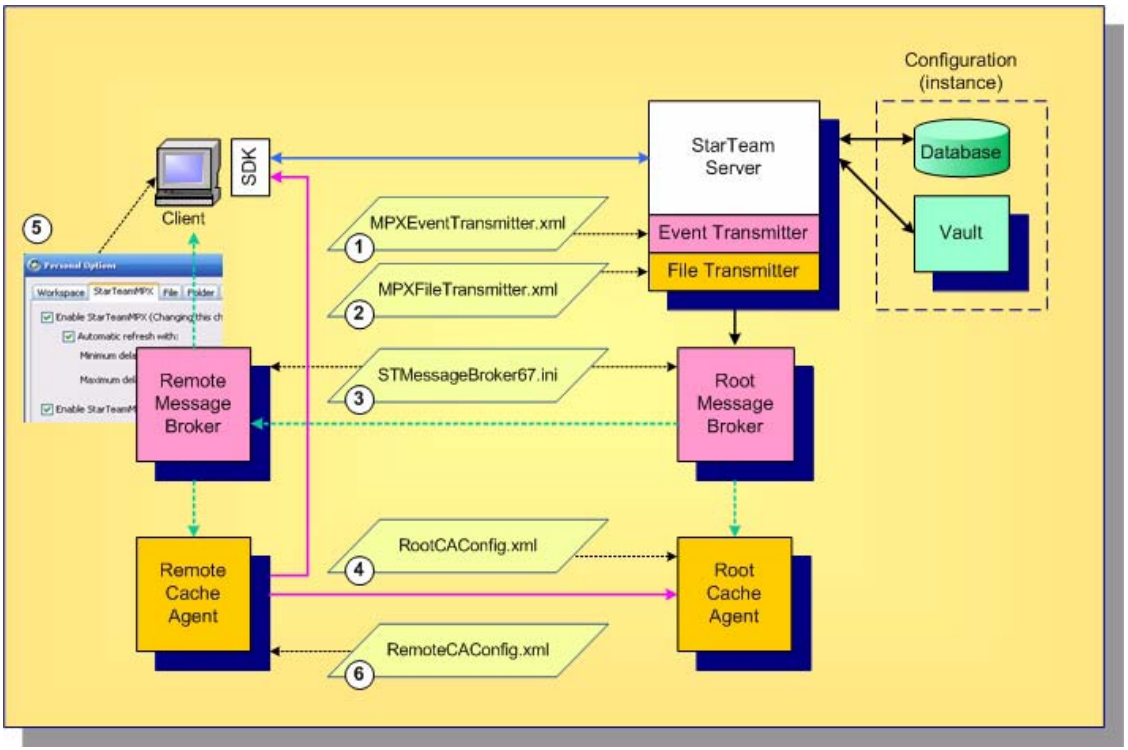
Object caching requires the following StarTeam components:

- **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 Cache Agents require an EA license.)
- **Root Cache Agent:** One MPX Cache Agent version 2009 or later must be configured as a “root” Cache Agent for each StarTeam configuration that uses object caching. Cache Agents always cache file content; object caching requires additional configuration. **When a pre-2009 StarTeam server is upgraded, the root Cache Agent for its StarTeam configuration(s) must be upgraded to the same release at the same time.**
- **Remote Cache Agents:** One or more MPX Cache Agents can be configured as “remote” Cache Agents in each desired location. A Cache Agent version 2009 or later is required to support object caching. Pre-2009 remote Cache Agents will interoperate with a 2009 or later root Cache Agent, but they will only support file caching. If a 2009 or later remote Cache Agent is configured to use a pre-2009 root Cache Agent as its “upstream” 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 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 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 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.

Figure 5.2 MPX Configuration Order



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 **Event Transmitter**. To enabled it, create a suitable MPXEventTransmitter.xml file and place it in the folder EventServices/<configuration> for each StarTeam configuration that you want MPX-enabled. For example, a typical folder location of this file for the StarDraw configuration on a Windows installation is: *C:\Program Files\Borland\StarTeam Server <version>\EventServices\StarDraw*

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 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 details on MPX profiles.

Next, enable the **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 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 EventServices/<configuration> 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 13.0 release. However, **be sure to start the StarTeam** 13.0 **Server for a given configuration at least once** before starting the 13.0 root 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 details.

④ Configuring the Root Cache Agent

There are several configuration options for enabling object caching for root Cache Agents. As before, a single root Cache Agent process services one StarTeam configuration. If multiple 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 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 details.

A root Cache Agent is designated by creating an XML configuration file (RootCAConfig.xml in Figure 5.1) 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 Cache Agent requires two alternate elements that provide it with more information about the StarTeam configuration that it services. These are shown below:

```
<?xml version="1.0" ?>
 <MPXCacheAgent>
     <RootCacheAgent>
         <ServerConfigsFile>C:\Program Files\Borland\StarTeam Server
             <version>\
             starteam-server-configs.xml</ServerConfigsFile>
         <ConfigName>StarDraw</ConfigName>
         <PreCharge>TipsOnly</PreCharge>
     </RootCacheAgent>
     ... other Cache Agent options ...
 </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 Cache Agent will service. From these two parameters, the root Cache Agent determines how to access the configuration's database, and it also determines the configuration's repository path.

At start-up, the root Cache Agent will open the database and "pre-charge" its local cache with any missing object revisions of configured types. By default, the root 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 Cache Agent operates on the same machine as the StarTeam Server that it services. When the root 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-server-configs.xml
  </ServerConfigsFile>

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

- The machine on which root Cache Agent operates must be able to access the configuration's database through the same database definition as the StarTeam Server. The root 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\StarDraw_SSE2005</ RootRepositoryPath>*

Finally, to enable object caching, the 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 Cache Agents, hence it is specified as a child element of the outer-most `<MPXCacheAgent>` group. There are two basic ways to specific which object types to cache. The most common way to enumerate each object type within a `<ObjectType>` element. Example:

```
<?xml version="1.0" ?>
<MPXCacheAgent>
    ...
    <CacheTypes>
        <ObjectTypes>
            <ObjectType>Change</ObjectType>
            <ObjectType>Requirement</ObjectType>
            <ObjectType>Task</ObjectType>
            <ObjectType>$CustomComponents$</ObjectType>
        </ObjectTypes>
    </CacheTypes>
    ...
</MPXCacheAgent>
```

In this example, the 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 Cache Agent will perform start-up pre-charging (as discussed above), catch-up and forwarding operations for upstream remote 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>
    ...
```

```
    <CacheTypes>
        <ObjectTypes AllTypes="True"/>
    </CacheTypes>
    ...
</MPXCacheAgent>
```

Note that 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 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 Cache Agent at one time, it is not efficient to read each file directly from disk. Instead, 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 Cache Agent uses for memory caching. When the 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 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 Cache Agents

To enable support object caching in a remote 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 Cache Agents as summarized below:

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

⑥ Enabling Object Caching for the Cross-Platform Client

Several StarTeam clients support the use of Cache Agents for file check-out operations, including the 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 Cache Agent to fetch object properties.

To enable object caching for the CPC:

1. Start the Cross-Platform Client.
2. Select Tools > Personal Options from the menu bar.
3. From the resulting Personal Options dialog, select the StarTeamMPX tab.
4. Select the Enable StarTeamMPX Cache Agent check box.
5. 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 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 Cache Agent for Each StarTeamMPX option button and let the client do the work.
6. You can change the number of threads in the Maximum Request Threads text box, but the default should be adequate for most users needs.
7. Under Use 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 Cache Agent when performance is estimated to be faster than going to the StarTeam Server.

Chapter 6

Configuring Clients

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

Note Unless otherwise stated, this chapter's references to client refer to only the StarTeam Cross-Platform client.

Using MPX from a Client

To configure support for StarTeamMPX on your workstation:

1. Start the client.
2. Select the Tools > Personal Options command from the menu bar.
3. In the resulting *Personal Options* dialog, select the StarTeamMPX tab.
4. Select the "Enable StarTeamMPX" check box to use StarTeamMPX with any MPX-enabled StarTeam Server connected to by the client.
5. Do one of the following:
   - To refresh manually (*Shift+F5*), clear the "Automatic refresh with" check box.
   - To 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" text box. The default is 5 seconds.

3 Set a maximum number of seconds between refreshes in the "Maximum delay of ___ seconds" text box. 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 StarTeamMPX:

- Clear the "Enable StarTeamMPX" check box to stop support for any 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 autorefresh setting, and (for StarTeamMPX configurations) whether support for StarTeamMPX is enabled.

Table 6.1 shows the icons and words that provide information about StarTeamMPX when they appear on the status bar.

Table 6.1 MPX Information in the Client's Status Bar

Status Bar Information	Description
Yellow lightning bolt	Indicates that StarTeamMPX 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 StarTeamMPX 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 StarTeamMPX 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 StarTeamMPX is not available for the currently selected project view.
Instant	Indicates that StarTeamMPX's autorefresh is turned on.

Table 6.1 MPX Information in the Client's Status Bar

Status Bar Information	Description
Auto	Indicates that your workstation's autorefresh is turned on—but that StarTeamMPX's autorefresh is either turned off or unavailable. (Your workstation's autorefresh option is on the Workspace tab of the *Personal Options* dialog.)
Manual	Indicates that your workstation's autorefresh is turned off and that StarTeamMPX's autorefresh 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 a StarTeamMPX connection profile other than the client default profile:

1. In the client, display the *Open Project Wizard* by selecting the Project > Open command.
2. Select the server configuration for which you wish to select a non-default client profile, and click Server Properties.
3. On the resulting *Server* dialog, click **MPX Profiles**.

   The resulting dialog lists each profile defined in the Event Transmitter XML file for this server configuration.
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, and then click **OK** on the *Server Properties* dialog.

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 MPX information in this log:

1. Start the client.
2. Select the Tools > Personal Options command from the menu bar.
3. In the resulting *Personal Options* dialog, select the Workspace tab.
4. Select the Log StarTeamMPX Events check box.
5. Click OK.

To review the StarTeam log, select Tools > StarTeam Log from the menu bar.

Working with the StarTeamMPXCommands.txt File

In some cases, you may want a specific client to use different message engine properties than any of those available in the Event Transmitter's XML file. One reason for wanting to do this is for external clients operating outside of the corporate firewall when no profile has been defined for external users. For such users, you may require that they use an external Message Broker that is not defined in any of the Event Transmitter's profiles.

To direct a client to use an external Message Broker, you need to set the server_names parameter for the appropriate Message Broker. This parameter, as well as other options, can be overridden by creating a file called StarTeamMPXCommands.txt and placing it in the same folder as the client executable (normally "C:\Program Files\Borland\StarTeam Cross-Platform Client <version>\" for the Cross-Platform client). This file must contain one parameter per line in the format:

setopt *optionName optionValue*

Valid parameter names and values are the same as the XML parameter names (without the angle-brackets) used within the Profile group of the MPXEventTransmitter.xml file. See the tables named "Event Transmitter Parameters" on page 36 and "Parameters for the Properties Group Inside the Profile Group" on page 38.

Note that the StarTeamMPXCommands.txt file uses one setopt command per line, whereas the MPXEventTransmitter.xml file uses an XML format. As an example, to override the server_names parameter for a client, the StarTeamMPXCommands.txt file would have to contain a line such as:

```
setopt server_names tcp:ExtHostA:5112
```

This parameter will set the server_names parameter that the client receives from the StarTeam Server, instructing it to use the Message Broker on host ExtHostA at port number 5112. Note that when a client opens a project on an

MPX-enabled server configuration, it always pings the Event Transmitter to ensure that it is visible within the Message Broker cloud. This ensures that if the client is using a different Message Broker than the Event Transmitter, they are properly connected into a cloud and visible to each other. If a client cannot successfully ping the Event Transmitter for a new project being opened, it will display an error dialog and proceed in non-StarTeamMPX mode.

The StarTeamMPXCommands.txt file can be used to override other Message Broker options as well, but only in unusual circumstances should anything other than the server_names parameter be overridden.

Using Cache Agent from the Cross-Platform Client and IDEs

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

Enabling Cache Agent Use

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

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 StarTeamMPX is not enabled, the Cache Agent cannot be auto-located: it must be configured with a specific address and port.

Server Properties

1. Start the Cross-Platform client.
2. Select Project > Open from the menu bar. The Open Project Wizard dialog box opens.
3. Select the server configuration for which you wish to enable Cache Agent use, and click Server Properties.
4. On the Server Properties dialog box, select the Cache Agent tab.
5. Under Use Cache Agent for, select whether the Cache Agent will be used for file caching (File Content) and/or 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 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 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 Cache Agent for Each StarTeamMPX 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.

Personal Options

1 Start the Cross-Platform client.

2 Select Tools > Personal Options from the menu bar.

3 From the resulting Personal Options dialog, select the StarTeamMPX tab.

4 Select the Enable StarTeamMPX 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 Cache Agent for, select whether the Cache Agent will be used for file caching (File Content) and/or 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 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 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 Cache Agent for Each StarTeamMPX 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.

Note From some IDEs, such as the Eclipse and Visual Studio .NET integrations, you can set StarTeam Personal Options although the menu command used to reach the options dialog may be different.

Checking out Files with Cache Agent

The visible advantage to using Cache Agent is the improved speed of file check-out operations. The more files you check out, the more advantage you will gain from Cache Agent. Over time, more and more of the files will come from 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 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 Cache Agent, by displaying the check-out statistics.

To monitor check-out statistics using Cache Agent:

1. Select the files to be checked out.
2. Select File > Check Out from the menu bar.
3. From the resulting Check Out dialog, 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 Cache Agent).

   After the operation completes, the Checkout Statistics dialog provides a summary.

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

Using Cache Agent with bco

You can use Cache Agent with the Bulk Checkout (bco) utility. For more information about this utility see the *StarTeam Cross-Platform Client Help* guide or the Cross-Platform Client online help.

Chapter 7

Running MPX Components

Most of the StarTeamMPX 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 Windows and Linux platforms. It also explains how to start and stop Cache Agent on those platforms.

Running Message Broker on Windows

On Windows platforms, the Message Broker is installed as a 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 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, choose Start > Settings > Control Panel > Administrative Tools > Services.On Windows XP systems, choose Start > Control Panel > Performance and Maintenance > Administrative Tools > Services.
2. Select the service named “StarTeam Message Broker 6.8”.
3. Click Start.

When the Message Broker starts, it reads the STMessageBroker68.ini configuration file. On Windows systems, this file is typically located in the “C:\Program Files\Borland\Message Broker <version>” 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” on page 26 for details on the contents of the STMessageBroker68.ini file.

Stopping a Message Broker

Under most conditions, a Message Broker runs continuously; however, occasionally 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 StarTeamMPX.

1. You should first notify users that StarTeamMPX will be unavailable.
2. On the computer where the Message broker is installed, choose Start > Settings > Control Panel > Administrative Tools > Services.On Windows XP systems, choose Start > Control Panel > Performance and Maintenance > Administrative Tools > Services.
3. In the Service list box, select the service named “StarTeam Message Broker 6.8”.
4. Click Stop.

Running a Message Broker on Linux

On Linux platforms, the Message Broker runs as a daemon and is incorporated into the init.d initialization and termination process. This means that the following file is placed in the /etc/init.d/ directory by the installer:

- STMB68: This file is the initialization/termination script for the Message Broker.

These script files are linked to the run level 3 multi-user state by placing the following files in the /etc/rc3.d/ directory:

- S98stmb68: This file is a symbolic link to “/etc/init.d/STMB68”, which is the Message Broker script. Because the file prefix is “S”, the OS calls this script with the parameter “start” when the system is entering run level 3. When called with this parameter, the script will start the Message Broker. The number “98” is the startup sequence number, which controls the order in which scripts are executed. (The sequence number can be changed to another value by renaming the file; see the man pages for “init.d” for more information.)
- K98stmb68: This file is a symbolic link to “/etc/init.d/STMB68”, which is the Message Broker script. Because the file prefix is “K”, the OS calls this script with the parameter “stop” when the system is exiting run level 3. When called with this parameter, the script will terminate the Message Broker.

As a result of these scripts, the Message Broker is normally started and stopped automatically as the system initializes and shuts down.

If you need to manually start or stop the Message Broker, you can do so by performing the following steps:

1. Log onto an account with root privileges.
2. Change directories to the folder containing the initialization/termination scripts. For example:

   ```
   cd /etc/init.d
   ```

3. Run the appropriate script with the parameter “start” to start the corresponding daemon, or “stop” to stop an existing daemon.
   - To start the Message Broker: STMB68 start
   - To stop the Message Broker: STMB68 stop

Running Cache Agents

To run the Cache Agents successfully, start the applications you have installed in the following order: Message Brokers, StarTeam Server (the server configuration starts the transmitters), Root Cache Agent, and Remote Cache Agent. For more details about starting MPX components see, “Dependencies - Startup Order for MPX Components,” on page 17.

Running Cache Agent On Windows

Cache Agents can be run as service or console applications. Up to 25 Root and Remote 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 StarTeamMPX Cache Agent and an internal name of CacheAgentService. You will need to register the others.

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

Running Cache Agent as a Service

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

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

You can also control the 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

Parameter	Description
configFile	The default configuration file is CacheAgentConfig.xml. If you use multiple configuration files, each one must specify unique values for CachePath ad RequestPort so that Cache Agent services do not interfere with each other's operation.
-dependson *list*	Specifies service dependencies for the new Cache Agent service. The *list* must be a quoted, space-separated list of internal (not display) names of the services on which the Cache Agent service will depend. (A service's internal name is its registry key within the Windows registry.) The most common dependency for a 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 "StarTeam Message Broker 6.8", but its internal name is typically "StarTeamMessageBroker6.8". Consequently, to make a Cache Agent depend on the Message Broker service on the same computer you would use: -dependson "StarTeamMessageBroker6.8"
-log *logFile*	Specifies a log file name other than the default, which is CacheAgentService.log. For more details about log file naming conventions, see "Log File" on page 83.
-name *serviceName*	Specifies the display name for the service in the Control Panel's Services utility. The default is StarTeamMPX Cache Agent. This is **not** the internal name which defaults to CacheAgentService (or, if that is already in use, CacheAgentService2 through CacheAgentService25).
-register	Register the service with the specified start mode (Manual or Auto), optionally with a specific configuration file at startup.

Parameter	Description
-start	Starts the service, optionally with a specific configuration file.
-unregister	Removes a service. For example, if you change the Cache Agent from Manual to Auto, you would unregister it and reregister it. A service must be stopped before it can be unregistered.
-verbose	Causes another more detailed log to be generated. It defaults to CacheAgentService-debug.log. For more details about log file naming conventions, see “Log File” on page 83.

Examples

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

```
CacheAgentService -register Auto c:\CAConfigs\Prod1RootCA.xml -
dependson "StarTeamMessageBroker6.8" –name "Prod1 Root CA" –verbose
```

As with the register command, the default service name is “StarTeamMPX 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 the Cache Agent service 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 Cache Agent is running.

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

Running Cache Agent As a Console Application

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

For this scenario, each 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**]

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

Log File

When the Cache Agent service runs, it creates a log file in its installation folder. The default file name for the log is CacheAgentApp.*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:
http://*host***:***port***/log**

host identifies the computer on which the Cache Agent is running

port provides its configured port number.

Running Cache Agent on Linux

To run a Cache Agent on a Linux system, use the following start command:
cacheagentapp -c RootCacheAgentConfig.xml -d start

To stop running a Cache Agent on a Linux system, use the following stop command:
cacheagentapp -c RootCacheAgentConfig.xml -d stop

Appendix

A

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 Borland support personnel.

This section shows sample StarTeamMPX-related messages that may appear in a server log file. These messages are typically prefixed with “StarTeamMPX”.

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 a StarTeamMPX 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 MPX-enabled server configuration. (The StarTeamMPX-specific messages appear in boldface type.)

*********Starting boot log*********
StarTeam Server version: X.X.XXX configuration: YourServerConfiguration
Collation: Latin1_General_CI_AS
Microsoft Windows XP Service Pack 2 (Build 2600)
DSN: YourServerConfigurationName, ODBC driver version: SQL Server 03.85.1117ODBC driver version: SQL Server 03.81.9041
Server database opened.
Server configuration database opened
The security manager loaded.
Server component loaded: AuditComponent
Server component loaded: ChangeRequestComponent
Server component loaded: FileComponent
Server component loaded: RequirementComponent
Server component loaded: TaskComponent
Server component loaded: TopicComponent
The project manager loaded.
The logon manager loaded.
StarTeamMPX: Connecting to project <starteam> on <tcp:YOURSERVER:5101> RTserver
StarTeamMPX: Using tcp protocol
StarTeamMPX: Message from RTserver: Connection established.
StarTeamMPX: Start subscribing to subject </StarTeamServer_be5ee3b0-c719-49c6-a1a1-f493764a03f5>
Loaded asynchronous handler StarTeamMPX Transmitter in Module C:\Program Files\Borland\StarTeam Server X\xmitt.dll.
StarTeamMPX File Transmitter: Using journal file: C:\YourServerConfiguration\CacheJournal.dat
StarTeamMPX File Transmitter: Scanning hive #10 for new files
StarTeamMPX File Transmitter: Hive #10 scan complete
StarTeamMPX File Transmitter: Scanning hive #11 for new files
StarTeamMPX File Transmitter: Hive #11 scan complete
5 named user licenses will be granted.
TCP/IP (Socket) protocol registered with the endpoint 49201
Server successfully initialized.

Reconnect Messages

If an MPX-enabled server configuration loses the connection to its Message Broker, it records that event in the server log file. In most circumstances, the server configuration attempts to re-establish communication with the Message Broker. 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 a StarTeamMPX configuration loses and then reconnects to its Message Broker.

```
StarTeamMPX: Lost connection to RTserver: error code = 10
StarTeamMPX: Waiting before reconnecting
StarTeamMPX: Attempting to reconnect to RTserver
StarTeamMPX: Connecting to project <YourProject> on
<tcp:YOURSERVER:5101> RTserver
StarTeamMPX: Using tcp protocol
StarTeamMPX: Could not connect to <tcp:YOURSERVER:5101> RTserver
StarTeamMPX: Skipping starting <tcp:YOURSERVER:5101> RTserver
StarTeamMPX: Cannot reconnect to RTserver
    .
    .
    .
StarTeamMPX: Attempting to reconnect to RTserver
StarTeamMPX: Connecting to project <YourProject> on
<tcp:YOURSERVER:5101> RTserver
StarTeamMPX: Using tcp protocol
StarTeamMPX: Message from RTserver: Connection established.
StarTeamMPX: Start subscribing to subject
</StarTeamServer_be5ee3b0-c719-49c6-a1a1-f493764a03f5> again
```

Appendix

B

Troubleshooting StarTeamMPX

To determine whether your StarTeamMPX system is operating correctly, you can perform the following steps. This chapter assumes you are troubleshooting on a Windows operating system.

To check your StarTeamMPX system:

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)
   - STMessageBroker68.ini (for each Message Broker)
   - RootCAConfig.xml (for each Root Cache Agent)
   - RemoteCAConfig.xml (for each Remote Cache Agent)
2. If they are not already running, start the Message Brokers.
3. For each Message Broker you start, start an 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 the StarTeamMPX is enabled for your workstation. See Chapter 6, "Configuring Clients" for details.
6. Enable client StarTeamMPX options:
   a. In your client, select the Tools > Personal Options command.
   b. In the resulting *Personal Options* dialog, select appropriate options on the *Workspace* and StarTeamMPX 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 selecting the Tools > StarTeam Log command. It also has settings for enabling and disabling MPX.

   Only the Cross-Platform client and IDEs based on StarTeam Cross-Platform and .NET components have options for Cache Agent.

   c. Click OK.

   Test these settings.
7. In your client, open a view from an MPX-enabled server configurations. If MPX is enabled in both the client and the server configuration, a yellow lightning bolt appears in the status bar.

Diagnosing a Message Broker

To assist with diagnosing problems, the Message Broker can generate logging information. To enable diagnostic tracing for the Message Broker, add the following lines to the STMessageBroker68.ini file:

```
setopt trace_file c:\temp\mbtrace.txt
setopt trace_flags timestamp
```

After adding these lines and saving the file, restart the Message Broker to begin generation of the diagnostic messages. The messages are written to the trace file, prefixed with time stamp information.

Note that there may be some delay between the time a trace message is generated and when it is written to the log file. Consequently, the most recent trace messages may not appear until the Message Broker is stopped, allowing the file to be closed.

Index

A

access rights
  enforced by caching modules 14
AutoLocate
  Cache Agent parameter 59
autorefresh status 74

B

bold convention 6
brackets convention 6
Bulk CheckOut (bco) command-line utility
  MPX usage 8

C

Cache Agent
  described 12
  logs 60
  planning considerations 48
  registering as service 82
  starting 84
  starting console application 84
  status 60
  stopping 84
  stopping console application 84
  viewing log file 83
Cache Agent parameter
  AutoLocate 59
  CacheCheckInterval 54
  CachePath 54
  CatchupCheckInterval 57
  ContentSource 58
  enable_control_msgs 56
  InboundAddresses 54
  InitialRequestThreads 55
  ListenAddresses 55
  MaxCacheSize 55
  MaxCatchupSize 55
  MaxConnections 55
  MessageBroker 56
  PrechargeSize 59
  Project 58
  Projects 58
  RequestPort 56
  RequestReadTimeout 57
  RootCacheAgent 60
  RootHiveIndexPath 60
  RootJournalPath 60
  RootRepositoryPath 60
  server_names 56
  server_start_delay 56
  ServerGUID 58
  SharePolicy 57
  socket_connect_timeout 56
  UpstreamCache 59
  UpstreamHost 59
  UpstreamPort 59
Cache Agent service
  starting 81
  stopping 81
cache messages
  routing between Message Brokers 22
  routing in unconnected clouds 22
cache refresh, setting the refresh timer 74
CacheCheckInterval
  Cache Agent parameter 54
CachePath
  Cache Agent parameter 54
caching modules in client
  enforcing user access rights 14
  explained 8
CatchupCheckInterval
  Cache Agent parameter 57
Client
  caching modules 8
client
  cache updates 22
  described 12
  item refresh operation 12
  opening projects on StarTeamMPX 21
  receiving updates from Message Brokers 21
  setting StarTeamMPX refresh timer 74
client default profile 35
ClientDefault
  MPXEventTransmitter.xml parameter 37
clouds
  defined 20
  message routing when unconnected 22
  volume considerations 23
communication, establishing between Message Brokers 21
configuration files, locating STMessageBroker68 21
configuring
  Remote Cache Agent 52
  Root Cache Agent 51
ContentSource
  Cache Agent parameter 58
conventions

bold 6
brackets 6
fixed-space type 6
italics 6
menu selections 6
vertical bars 6

D

dependencies
order of installation and startup of MPX components 17
developer support 5
DMZ *See* firewalls
documentation conventions 6

E

enable_control_msgs
Cache Agent parameter 56
MPXEventTransmitter.xml parameter 40
Event Transmitter
described 12
startup process 36
XML file locations 31

F

file transmission 13
File Transmitter
described 12
startup process 44
XML file locations 31
files
server log 85
STMessageBroker68.exe 21
STMessageBroker68.ini 21
FileTransmitter.xml 32
general format 45
FileTransmitter.xml parameter
JournalPath 45
JournalTrimInterval 45
MaxJournalAge 45
FileTransmitterTemplate.xml 32
firewalls 24
fixed-space type convention 6

I

icons in client status bar 74
IDEs
MPX usage 8
InboundAddresses
Cache Agent parameter 54
.INI files, STMessageBroker68.ini 21
InitialRequestThreads
Cache Agent parameter 55
installation dependencies 17
italics convention 6
item refresh operation 12

J

JournalPath
FileTransmitter.xml parameter 45
JournalTrimInterval
FileTransmitter.xml parameter 45

L

ListenAddresses
Cache Agent parameter 55
logs
Cache Agent 60, 83

M

MaxCacheSize
Cache Agent parameter 55
MaxCatchupSize
Cache Agent parameter 55
MaxConnections
Cache Agent parameter 55
MaxJournalAge
FileTransmitter.xml parameter 45
menu selections convention 6
Message Broker
changing the endpoint 26
cloud configuration 25
clouds defined 20
configuration file 21
configuring 25
creating clouds 21
described 12
executable file 21
external users 24
fault-tolerant operation 24
firewalls 24
installing on different computer from StarTeam Server 23
installing on same computer as StarTeam Server 23
large number of simultaneous users 23
maximum number of simultaneous users 23
maximum number of TCP/IP connections 23
planning considerations 19
routing messages between 22
setting up communication between 21
starting 79
STMessageBroker68.exe 21

STMessageBroker68.ini 21
stopping 79, 80
UDP protocol 71
updating clients 21, 22
using just one 23
WAN usage 24
Message Broker profiles
described 14
MessageBroker
Cache Agent parameter 56
MPX components
dependencies 17
MPX-enabled server configuration
described 11
MPXEventTransmitter.xml 32
general format 37
sample unicast profile 41
MPXEventTransmitter.xml parameter
ClientDefault 37
enable_control_msgs 40
Profile 38
project 39
Properties 38
server_keep_alive_timeout 40
server_names 39
server_read_timeout 40
server_start_delay 40
server_start_max_tries 40
server_write_timeout 40
ServerDefault 37
MPXEventTransmitterTemplate.xml 32

P

PrechargeSize
Cache Agent parameter 59
product support 5
Profile
MPXEventTransmitter.xml parameter 38
profiles
client default 35
described 14, 35
multiple connections 43
non-default 75
server default 35
Project
Cache Agent parameter 58
project
MPXEventTransmitter.xml parameter 39
Projects
Cache Agent parameter 58
Properties
MPXEventTransmitter.xml parameter 38
publish/subscribe communication 7

R

refresh timer, setting for client 74
registering
Cache Agent service 82
Remote Cache Agent
configuring 52
described 12
example configuration file 52
RequestPort
Cache Agent parameter 56
RequestReadTimeout
Cache Agent parameter 57
Root Cache Agent
configuring 51
described 12
example configuration file 51
RootCacheAgent
Cache Agent parameter 60
RootHiveIndexPath
Cache Agent parameter 60
RootJournalPath
Cache Agent parameter 60
RootRepositoryPath
Cache Agent parameter 60

S

scalability of StarTeamMPX 8
security
data encryption 13
StarTeamMPX 13
user access rights 13
user authentication 13
server configuration
described 11
server default profile 35
server log file
reconnect messages for StarTeamMPX 87
sample entries 85
start-up messages for StarTeamMPX 86
Server Properties dialog, using to set encryption level 13
server_keep_alive_timeout
MPXEventTransmitter.xml parameter 40
server_names
Cache Agent parameter 56
MPXEventTransmitter.xml parameter 39
server_names parameter, defined 21
server_read_timeout
MPXEventTransmitter.xml parameter 40
server_start_delay
Cache Agent parameter 56
MPXEventTransmitter.xml parameter 40

server_start_max_tries
MPXEventTransmitter.xml parameter 40
server_write_timeout
MPXEventTransmitter.xml parameter 40
ServerDefault
MPXEventTransmitter.xml parameter 37
ServerGUID
Cache Agent parameter 58
SharePolicy
Cache Agent parameter 57
socket_connect_timeout
Cache Agent parameter 56
square brackets convention 6
StarTeam clients
displaying MPX status 74
logging MPX information in StarTeam.log 76
overriding profiles 76
using MPX 73
using profiles 75
StarTeam Cross-Platform client
checking out files with Cache Agent 78
MPX usage 8
StarTeam Cross-Platform clients
enabling Cache Agent use 77
StarTeam Server
described 11
StarTeam server configuration
described 11
StarTeam.log
logging MPX information 76
StarTeamMPX
data encryption 13
disabling support in client 74
scalability 8
security 13
troubleshooting 89
user access rights 13
user authentication 13
StarTeamMPX Event Transmitter
described 12
StarTeamMPX File Transmitter
described 12
StarTeamMPXCommands.txt 76
starting
Cache Agent 84
Cache Agent service 81
Message Broker 79, 80
starting console application
Cache Agent 84
startup dependencies 17
status
Cache Agent 60
status bar icons showing MPX info 74
STMessageBroker68
described 21
server_names parameter 21
STMessageBroker68.exe 21
STMessageBroker68.ini
editing 26
stopping
Cache Agent 84
Cache Agent service 81
Message Broker 79, 80
support for StarTeamMPX 74
stopping console application
Cache Agent 84
support 5
syntax conventions
mutually-exclusive choices 6
options 6
variables 6

T

TCP/IP, maximum number of Message Broker connections 23
technical support 5
troubleshooting
StarTeamMPX 89

U

UDP protocol, used by Message Brokers 71
unicast profile
described 35
UpstreamCache
Cache Agent parameter 59
UpstreamHost
Cache Agent parameter 59
UpstreamPort
Cache Agent parameter 59
Using Cache Agent from the Star-Team Cross-Platform Client 77

V

vertical bars convention 6
volume considerations 23

W

Web Edition
MPX usage 8