MICRO FOCUS

StarTeam 16.1

Cross-Platform Client Help

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

2017-05-01

Contents

StarTeam Cross-Platform Client

Welcome to the StarTeam Cross-Platform Client

Welcome to StarTeam
Tour of the UI
Licensing for StarTeam
Logging On to and Off of a Server
Quick Access to Projects and Items

Online resources

Micro Focus Learning Center
Micro Focus Infocenter
Micro Focus SupportLine
Micro Focus Product Updates
Micro Focus Community Forums
Micro Focus Training Store

Provide feedback

Contacting Support
Email us feedback regarding this Help

Welcome to StarTeam

StarTeam is a software change and configuration management solution designed to meet the needs of local and distributed teams regardless of size and work style. Team members can work whenever and wherever they like and benefit from integrated change management, defect tracking, file versioning, requirements management, and project and task management capabilities for flexible project control. StarTeam is a robust platform for coordinating and managing the entire software project throughout the software development life-cycle.

Introduction

This section provides introductory information about StarTeam.

Installation and Licensing for StarTeam

Installation

Installation instructions for installing StarTeam products can be found in the *StarTeam Installation Guide*.

Licensing

StarTeam is available in three licensing packages:

Enterprise StarTeam Enterprise provides a basic feature set, including the StarTeam Server, StarTeamMPX (MPX Event Transmitter and MPX Message Broker), the Cross-Platform Client, StarTeam Web Client, LDAP QuickStart Manager, and the SDK. The requirements component is not available with this license, however, it does provide access to custom fields.

Enterprise Advantage StarTeam Enterprise Advantage has all the StarTeam Enterprise features plus the Requirement component, StarTeamMPX (MPX Cache Agent and MPX File Transmitter), and StarTeam Workflow Extensions which include alternate property editors (APEs) that enable you to create custom forms and design workflow rules to control how all the items in a component move from state to state. StarTeam Datamart is available for purchase.

Products Included with StarTeam Enterprise Licenses

The following provides a summary of StarTeam products that come with the StarTeam Enterprise license. The installation instructions for some products are not in this consolidated installation guide, but are located in the respective guide of that product and are noted.

StarTeam Server A StarTeam Server stores artifacts (files, change requests/defects, tasks, and topics) for StarTeam clients. A server can support one or more server configurations on the same computer. Install StarTeam Server on a computer that is accessible to all users.

MPX Message Broker Pushes information from the StarTeam Server to its clients. Usually an administrator sets up a cloud of Message Brokers to improve server performance for users in diverse geographic locations. One (sometimes two) root Message Brokers are set up for the server, usually on the same computer or in a network-near location. For Express, MPX is probably not needed.

StarTeam Cross-Platform Client The StarTeam Cross-Platform Client is the most used client and provides users with access to all of the artifacts on the server. The Cross-Platform Client is a pure Java client that provides support of operating systems where a compatible JRE or JDK are available. As such, Cross-Platform Client is available for the Microsoft Windows, Solaris, and Linux operating systems.

StarTeam Visual Studio Plugin The StarTeam Visual Studio Plugin provides the StarTeam software configuration management capabilities tightly integrated with the Microsoft Visual Studio development environment. Using this integration makes it possible for you to develop applications in the Microsoft Visual Studio environment while simultaneously using the version control, change request, topic, task, and requirement component assets of StarTeam. The integration brings StarTeam main menu commands, context menu commands, and an embedded StarTeam client (providing much of the same look-and-feel as the full-featured Cross-Platform Client) to the Microsoft Visual Studio development environment.

StarTeam Eclipse Plugin StarTeam Edition for Eclipse allows you to share projects on StarTeam Server and projects in the Eclipse workspace, but it is much more than just a version control plug-in. This integration offers project teams a customizable solution providing requirements, task, and change management, defect tracking and threaded discussions tightly integrated within the Eclipse platform.

StarTeam Web Server and StarTeam Web Client The StarTeam Web Server makes it possible for users to access the server from their browsers using the StarTeam Web Client. The StarTeam Web Client is an intuitive web-based interface that many simultaneous users can use to connect to one or more StarTeam Servers to access projects and manage items. This product contains a core feature set designed to meet the needs of users responsible for viewing, creating, and editing StarTeam change requests, requirements, tasks, and topics. Browsing files and a limited set of file operations are also available.

Note: You must have a StarTeam user license to use the Web Client.

LDAP Quickstart Manager

The StarTeam Server can provide password authentication via a directory service, such as LDAP Quickstart Manager (QSM) to add users to the server, along with their distinguished names (DN) (needed for authentication) and other user information.

Layout Designer

Use Layout Designer to create forms for artifacts, such as change requests. This allows you to put the most important properties on the first tab, etc. With the web client and an Enterprise Advantage server, a Layout Designer form works with workflow. This is not true of the StarTeam Cross-Platform Client where Layout Designer's use is only for form building.

This product is translated into English, French, German, and Japanese.

StarTeam SDK

The StarTeam SDK is cross-compiled so that it can be offered both as Java and .NET applications. The full SDK is used by developers to create additional applications that use the StarTeam Server.

Usually, the StarTeam SDK runtime is installed with clients automatically so it can be used by them to access the StarTeam Server. Occasionally, you may need to install the runtime.

StarTeam SCC Integration

The StarTeam SCC Integration works with any application that uses the Microsoft Source Code Control (SCC) Application Programming Interface (API). This API, originally designed by Microsoft to allow applications to work with Microsoft Visual SourceSafe, enables you to perform version control operations, such as checking files in and out, using StarTeam as the SCC provider.

StarTeam Quality Center Synchronizer

This product is available with all licenses.

StarTeam Quality Center Synchronizer can ensure that the same data appears in Quality Center and a database used by StarTeam Server. The goal of the synchronization is to provide access to the latest information about defects, whether the defects are being processed from Quality Center or from StarTeam. You can use Quality Center to add defects, and you can use StarTeam to indicate that those defects have been fixed and vice versa. Team members do not need to be aware of where the defect was last processed. The latest data is available at all times, as long as the databases are synchronized frequently.

StarTeam Microsoft Project Integration

Available with all licenses.

The interaction of the StarTeam Microsoft Project Integration and Microsoft Project make the jobs of both project planners and team members easier. Project planners use Microsoft Project to list the tasks that workers must perform. After exporting the tasks to StarTeam, they can gather information about the work accomplished by each team member in StarTeam, rather than communicating individually with each team member.

StarTeam Toolbar Utility

The StarTeam **Toolbar Utility** (Toolbar) is a component of the StarTeam designed to make it easier for you to log on to multiple servers and to launch different programs. It automatically caches the user name and password used to log on to each StarTeam or Caliber server, reducing the number of times that you must enter your logon information. The Toolbar is initially populated with shortcuts for the tools of the StarTeam and Caliber products that are installed on your workstation. Because the Toolbar uses the standard Microsoft Windows program shortcut feature, you can easily add any other program as a tool.

File Compare/ Merge File Compare/Merge is a graphical compare/merge tool delivered with the StarTeam Cross-Platform Client. It enables you to compare a file dynamically with the file in the repository, and manually or automatically merge the content of the two files. File differences are highlighted in the File Compare/Merge panes using a configurable color scheme, and action buttons display in the highlighted areas to simplify the merging process.

View Compare/ Merge View Compare/Merge is a comprehensive tool for comparing and merging views available with the StarTeam Cross-Platform Client. There are two versions of View Compare/Merge:

- **Graphical** Provides interactive comparison and merging with per-item and per-folder interaction, allowing you to carefully control which items are compared and how each difference is resolved
- **Command-line** Enables batch/shell-directed sessions.

Products Included with StarTeam Enterprise Advantage Licenses

In addition to the products included with StarTeam Enterprise licenses, StarTeam Enterprise Advantage licenses also include the products listed below. The installation instructions for some products are not in this consolidated installation guide, but are located in the respective guide of that product.

MPX Cache Agent A root MPX Cache Agent monitors the server's repository for file content and object properties. Via Message Broker, the data is pushed to remove MPX Cache Agents that are network-near to members of dispersed teams, improving the speed with which users access the data they need.

StarTeam Extensions StarTeam Extensions enables clients to take advantage of workflow and custom toolbar applications. The StarTeam Extensions files must be checked into the StarFlow Extensions project on each server configuration. If there is no StarFlow Extensions project, you need to create one.

StarTeam Extensions also provides API documentation and samples.

StarTeam Workflow Designer Use the StarTeam Workflow Designer to create workflows for specific artifact types (such as change requests/defects) per project or even per view.

StarTeam Notification Agent The StarTeam Notification Agent runs on the same computer as the StarTeam Server (or on a network-near computer) so that it can monitor the server and send notifications set up in your workflow.

Search Search allows users to perform ad hoc queries across servers and projects. The query results reflect the access rights of the user logged on to Search so information is shared across the organization without compromising security.

Datamart* StarTeam Datamart is a complementary product to the StarTeam Server. StarTeam Datamart uses the StarTeam SDK to communicate with the StarTeam Server to create a reporting database that you can use with popular third party reporting applications such as Crystal Reports and Business Objects (reporting applications are not included with StarTeam Datamart). StarTeam Datamart extracts data from a StarTeam Server and places the data into a relational database, where reporting tools can access it. StarTeam Datamart can extract information from every project, every view in each project, every folder in each view, and every item in each folder, and labels, links, and history for each item. You can restrict extraction of data to a particular project and view or only extract certain tables.

Datamart stores the data in any StarTeam supported database..

The product comes with both an Extractor (for an initial retrieval) and with a Synchronizer to update existing data sets.

TeamInspector*

TeamInspector is a continuous integration build server and build inspection tool. It works with StarTeam, Subversion, Perforce, and ClearCase. It requires the use of a database: Microsoft SQL Server 2005 SP3, Oracle 10g Release 2 version 10.2.0.4, or Apache Derby 10.4.2.0 or later.

Rhythm*

Rhythm is an Agile project tracking tool designed to allow you to:

- Organize, prioritize, and manage your Agile teams' backlogs.
- Plan your sprints, task out the work, and then track progress throughout the sprint.
- Get comprehensive visibility of all your Agile assets.

* Can be purchased separately and added to the Enterprise package.

About Source Control

Source Control Basics

Each source control system consists of one or more centralized repositories and a number of clients. A repository is a database that contains not only the actual data files, but also the structure of each project you define.

Most source control systems adhere to a concept of a logical project, within which files are stored, usually in one or more tree directory structures. A source control system project might contain one or many IDE-based projects in addition to other documents and artifacts. The system also enforces its own user authentication or, very often, takes advantage of the authentication provided by the underlying operating system. Doing so allows the source control system to maintain an audit trail or snapshot of updates to each file. By storing only the differences, the source control system can keep track of all changes with minimal storage requirements. When you want to see a complete copy of your file, the system performs a merge of the differences and presents you with a unified view. At the physical level, these differences are kept in separate files until you are ready to permanently merge your updates, at which time you can perform a commit action.

This approach allows you and other team members to work in parallel, simultaneously writing code for multiple shared projects, without the danger of an individual team member's code changes overwriting another's. Source control systems, in their most basic form, protect you from code conflicts and loss of early sources. Most source control systems give you the tools to manage code files with check-in and check-out capabilities, conflict reconciliation, and reporting capabilities. Most systems do not include logic conflict reconciliation or build management capabilities.

Commonly, source control systems only allow you to compare and merge revisions for text-based files, such as source code files, HTML documents, and XML documents. StarTeam stores binary files, such as images or compiled code, in the projects you place under control. You cannot, however, compare or merge revisions of binary files. If you need to do more than store and retrieve specific revisions of these types of files, you might consider creating a manual system to keep track of the changes made to such files.

Repository Basics

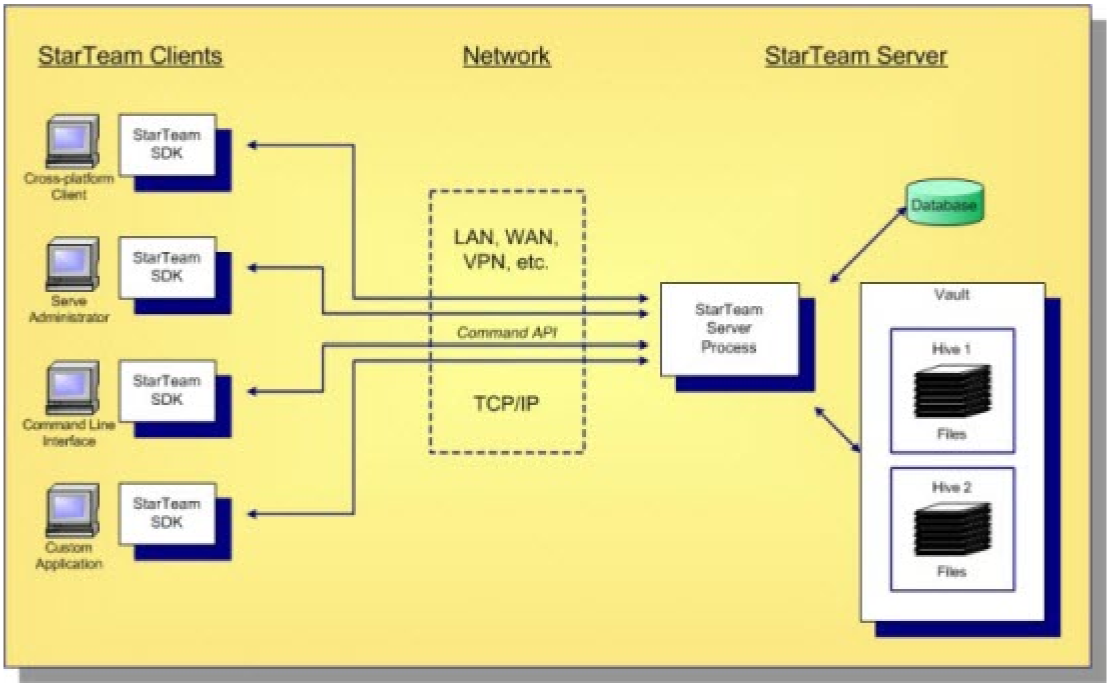
Source control systems store copies of source files and difference files in some form of database repository. In some systems, such as CVS or VSS, the repository is a logical structure that consists of a set of flat files and control files. In other systems, such as StarTeam, the repositories are instances of a particular database management system (DBMS).

Repositories are typically stored on a remote server, which allows multiple users to connect, check files in and out, and perform other management tasks simultaneously.

With StarTeam, you create a server configuration to identify a repository for StarTeam projects. Each server configuration acquires its own set of projects as they are created. The Server can run any number of server configurations. Because each server configuration must use a database, you need to make sure that you establish connectivity not only with the server, but also with the database instance.

Standard StarTeam Architecture Overview

The standard architecture represents the minimal components present in a StarTeam instance: a StarTeam Server process managing a vault and a database and one or more StarTeam clients. With just these components, all basic StarTeam functionality is available. The core components of the standard StarTeam architecture are depicted below.



StarTeam employs a client/server architecture. The StarTeam Cross-Platform Client, **Server Administration** tool, and StarTeam Command Line Tools are examples of bundled StarTeam clients. StarTeam clients use the freely available StarTeam SDK, so you can write custom applications that have access to the same features as the bundled clients. The SDK is fully featured in Java, .NET, and COM, allowing you to write custom applications for any environment. A single StarTeam client can have multiple sessions to any number of StarTeam Servers.

All StarTeam clients connect to a StarTeam Server process using TCP/IP, so virtually any kind of network can be used: LAN, WAN, VPN, or the public Internet. StarTeam uses a proprietary protocol called the *command API*, which supports compression and multiple levels of encryption. The command API has been optimized to support high performance, automatic reconnect, delta check-out for slow connections, and other important features.

A single deployment instance of StarTeam is known as a *server configuration*, usually shortened to just *configuration*. The persistent data of a configuration consists of a database and a *vault* and is managed by a single StarTeam Server process. The database holds all metadata and non-file artifacts, whereas file contents are stored in the vault. The database can be any of the supported databases and it can reside on the same machine as the StarTeam Server process or a separate machine. The StarTeam database and vault can be backed-up dynamically, while the server is in use. This supports 24 x 7 operations that want to minimize down time.

StarTeam’s vault is a critical component that affects performance and scalability. In contrast to the traditional delta storage technique, StarTeam’s vault uses an innovative architecture designed for scalability, performance, high availability, and dynamic expandability. Today, customers are storing up to a terabyte of data in a single StarTeam vault, but it was designed to store content up to a petabyte and beyond.

Within the vault, files are stored in containers known as hives. A hive is a folder tree containing archive and cache files on a single disk volume. Hives can be dynamically added on existing or new disk volumes, thereby allowing virtually unlimited capacity. StarTeam stores each file revision in a separate archive file in a manner that minimizes space usage as well as duplicate content. StarTeam’s vault uses less space than delta-based storage. In certain cases where it is more economical to send file deltas to clients instead of full versions, StarTeam generates and caches delta files. However, in most cases sending full versions is more economical.

Contacting Support

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

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

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

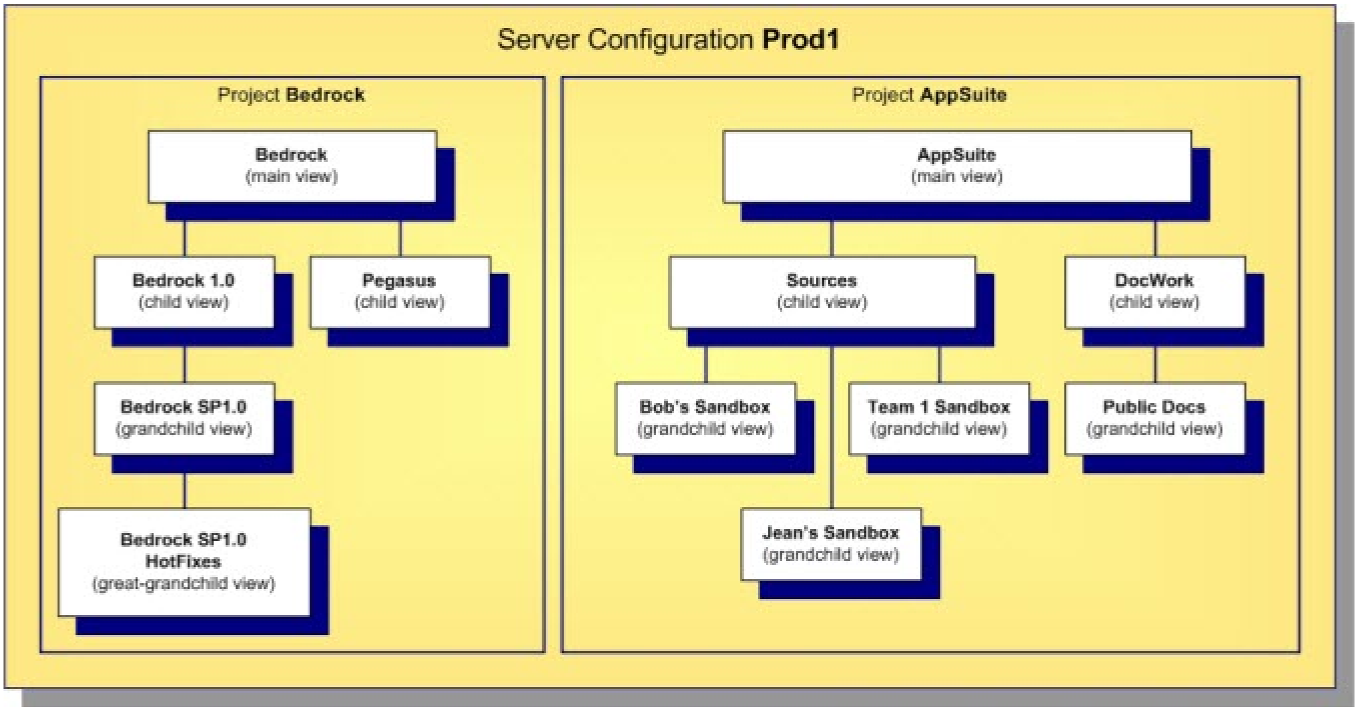
For support on third-party tools or documentation, contact the vendor of the tool.

StarTeam Basics

The topics in this section provide an overview of StarTeam concepts.

Containers

StarTeam Server configurations, projects, and views are *containers* that allow you to organize artifacts based on application, module, business unit, or other criteria. These three basic containers are illustrated in the diagram below:


Server Configuration **Prod1**
Project **Bedrock**
Bedrock (main view)
Bedrock 1.0 (child view)
Pegasus (child view)
Bedrock SP1.0 (grandchild view)
Bedrock SP1.0 HotFixes (great-grandchild view)
Project **AppSuite**
AppSuite (main view)
Sources (child view)
DocWork (child view)
Bob's Sandbox (grandchild view)
Team 1 Sandbox (grandchild view)
Public Docs (grandchild view)
Jean's Sandbox (grandchild view)

Server Configurations

A *server configuration* is also referred to as a repository or as an instance. All files, change requests, and other artifacts that can be interrelated and managed as a whole reside in the same configuration.

Note: Throughout the documentation, the terms *server configuration* and *server* are also used interchangeably. This is because each server configuration is often deployed on its own server machine, managed by its own StarTeam Server process. However, be aware that StarTeam allows multiple server configurations and server processes on a single machine, so the server configuration-to-machine relationship does not have to be one-to-one.

Projects

Within a server configuration, artifacts are organized into *projects*, which group and manage related items hierarchically in a set of folders. Creating a project allows you to put files under version control, set requirements, track change requests, manage tasks, audit user actions, and discuss the project. Each project has at least one view, called the initial or root view. For example, a project for a software product might include files on the product's functional specifications, marketing requirements, source code, and test suites, all stored in separate folders in the initial view. As the product progresses from one release to another, additional views of these folders can be created. One view could represent the 1.0 version of the product, while a second view represents the 2.0 version, and so on.

Before a server configuration can be used, at least one project must be created. A server configuration can hold multiple projects, each oriented to the lifecycle needs of a specific team, application, or component. The configuration in the diagram above has two projects: `BedRock`, perhaps for foundation components, and `AppSuite`, which could be used for applications belonging to a common suite.

Views

Each project consists of one or more *views*. Think of a view as a *subproject*. It is a subset of the project's contents that support a specific activity. Every project automatically receives a *main view* through which folders, files, and other objects can be organized. Additional *child views* can be created to represent subsets of the main project information, historic snapshots of project information, or work areas for new development work. StarTeam provides a wide range of options for view creation to support a wide range of development scenarios.

Workspaces

In addition to the three basis containers, StarTeam supports a client-side container called a *workspace*. A workspace is a folder hierarchy located on your computer or in your personal directory on a shared file server. However, the project does not have to exactly match your working folder and its child folders. For example, you may omit child folders in the working folder from a project or copy only specific child folders in an existing project to the working folder. When you add or check in files, the application copies the files from the working folder into the repository. When you check files out, the application copies the files from the repository into the working folder.

In addition to providing a well-defined area for check-in/check-out operations, a workspace allows StarTeam to compute the status of each file: which ones have been modified since they were checked-out, which ones are out-of-date, and so forth.

Artifacts

A typical software development lifecycle requires the development, evolution, and management of things other than source files such as requirements, models, graphics, change requests, schedules, tests, and so on. The term *artifact* refers to the generalization of objects that can be versioned, branched, merged, etc. StarTeam supports non-file artifact types directly, providing type-specific behavior for storage, versioning, merging and so forth. All artifacts are versioned, and some are branchable.

The built-in artifact types supported by StarTeam are summarized below:

Folder

Every view has one *root folder*, which typically has a tree of subfolders beneath it. Folders are patterned from the file system concept of directories. In many cases, you will want to create StarTeam folders that mirror specific directory structures. However, StarTeam folders can hold any kind of artifact, not just files. This concept may seem strange at first, but when you discover that you can organize change requests, tasks, and other non-file artifacts the same way you organize files, you will find this feature very powerful. Folders can branch, allowing the same folder to have different properties in each branch.

File

StarTeam allows you to store any kind of file: text or binary, authored or generated, small or very large. A few more features are provided for text files such as *keyword expansion* and *EOL conversion*, but all file types otherwise are treated identically. StarTeam allows single file revisions larger than 4GB. Files are branchable, allowing parallel version streams and merging.

Change Request

A *change request* (CR) is a general artifact that can represent a defect, enhancement request, or another reason for a software change. Because CRs are often the center of change management, the CR type is frequently extended with custom fields, custom GUI forms, and workflow rules. CRs can branch, allowing parallel modifications to the same CR for separate activities such as fixing the same defect in multiple releases. Using integration tools, you can import CRs from and keep them synchronized with other defect management systems.

Task

StarTeam *tasks* are modeled after project management tasks: they can be arranged hierarchically to represent task decomposition, they can be connected with predecessor/successor relationships, and they can be updated with progress units known as *work records*. You can import tasks from a project management system such as Microsoft Project, update and maintain them via StarTeam, and then synchronize them back to the original project source. In StarTeam, tasks are versioned but they do not branch.

Topic

A *topic* is very similar to a newsgroup message. Like newsgroup messages, topics can be organized into conversation threads. Because topics are artifacts, they are versioned (but not branched) and are stored in the repository with other artifacts. This allows you to capture more application lifecycle knowledge such as important discussions related to a design decision or a requirement approval.

Requirement If you do not have a formal requirements management (RM) tool, StarTeam requirements provide a convenient, lightweight artifact with which requirements can be captured. Requirements can be arranged hierarchically to represent decomposition, and they can be linked to other artifacts. Since requirements are independently-versioned artifacts, they are more accessible than requirements buried in documents, which are versioned at the whole-document level. If you use a requirements management system such as Caliber, those “formal” requirements can be imported as StarTeam requirements and organized together with other lifecycle artifacts. We provide integration tools to import, synchronize, and even link artifacts between StarTeam and Caliber. Requirements do not branch.

Audit An *audit* is a read-only *change event* artifact that is automatically generated for other artifact changes: add, modify, delete, move, label attach, link, etc. Because audits are automatically generated and immutable, they are not really artifacts per se, but StarTeam allows you to access them with similar GUI and SDK techniques as other artifacts, so you can think of them as read-only artifacts. The generation of audits and the length of time that they are retained are configurable.

These artifacts are all bundled with StarTeam, however you are not obligated to use them all. The code for each artifact type is encapsulated in a dynamically-loaded plug-in module called a *server-side component* (SSC). Each SSC is a code library suffixed with `.ssc` that resides in the StarTeam Server’s installation directory. If you rename an `.ssc` module before the server starts, the corresponding artifact type will not be used. For example, if you want to use StarTeam as a VCS only, just rename all `*.ssc` modules except for `file.ssc`.

Note: You always get folders, so there is no `.ssc` module for it. Also, we recommend you keep `audit.ssc` due to the value of the *change log* represented by audit artifacts.

Artifacts Versus Items

The difference between artifacts and items is that you can only access and update artifacts through items. Since StarTeam blends item and artifact properties into a single object (both graphically and in the SDK), you may think of them as a single concept. Although either term works equally well in most cases, we usually use the term *item* when we mean to include folder/view context that items add to artifacts. When the context is not important to the discussion, we use the term *artifact*.

The most important things to know about items are summarized below:

Artifacts can only be accessed through items With StarTeam, you can only fetch or update an artifact by directing your request to a specific item. There are no commands to directly access an artifact independent of an item. This means that all artifact access is influenced by the context of the associated item such as its parent folder and the view in which it lives.

Items form folder trees *Paths* are formed by each view’s *item tree*. This means that folder artifacts do not define their contents. Instead, what appears inside a folder is determined by the items that refer to the folder’s item as their parent. You don’t really move artifacts—you move items. Moving an item from one folder to another causes the item’s parent to be modified—the artifact referred to by the item isn’t touched. Under the hood, items are versioned similarly to artifacts. This means that changes such as moving an item to a new folder really creates a new item revision, causing the previous item to become historic.

Items facilitate sharing Items allow an artifact to appear in multiple folders, views, and projects. To make an artifact, including its entire history, appear in a new location, we only have to create a new item, which is pretty cheap. Sharing is analogous to “hard links” used in UNIX file systems.

Items influence An item has properties that control what artifact revision is referenced and how updates through the item are handled. Items store an OID that determines what artifact branch is

version behavior referenced. An item also stores a *configuration timestamp* to indicate whether it *floats* to the tip revision or is *pinned* to a specific revision of the referenced branch. An item's **branch-on-change** (BOC) flag indicates if the referenced artifact should branch when modified through the item. For example, if an item currently refers to artifact revision `1.7`, and **branch-on-change** is true, and an update is directed at the item, the artifact is branched by storing the updates to a new revision identified as `1.7.1.0`. Additionally, the item is modified to point to the new branch (since it has a new OID), and its BOC flag is set to `false`. Note that **branch-on-change** cannot be true for items that point to artifacts that can't branch (such as topics). Also, an item with **branch-on-change** equal to false and a pinned configuration timestamp is read-only because we can't update a historic revision and we can't start a new branch!

Items create promotion trees This is an advanced concept, so we'll just touch on it briefly here. Items that are shared to a new location "remember" the item from which they were shared. This "share parent" relationship is different than the "containment parent" relationship that forms item paths. It facilitates a concept called *automatic promotion*.

Folders

The project or server administrator usually creates projects and project views. If you are a typical user, you routinely open a particular project view and manage *your* folders and their contents, such as files and change requests. Managing application folders is very similar to managing a project. You can create folders, delete folders, and modify their properties—if you have the correct access rights.

Folder Hierarchy

When you create projects, you typically select locations on your workstations as the working folders for those projects. The working folder designated for a project also becomes the working folder for the project's root view and for the root folder in that view's folder hierarchy.

StarTeam treats folders as both containers and items. You can group items within a project view by placing them into folders. For example, a folder named *Source Code* can contain source code files and requested changes to those files. You can create folders automatically when you create a project, or add folders after you create the project. Project or server administrators (or team leads – this all depends on your organization) usually create projects, but anyone can create projects if they have the correct access rights. See your server administrator if you have questions regarding the access rights assigned to you.

When you create a project, StarTeam automatically creates the parent or root folder for that project at the same time. It is actually the root folder of the project's root (or initial) view. The project, view, and this root folder initially have the same name (although those names can be changed).

Usually, the user who creates a project sets up a hierarchy of folders on a workstation before creating the project. The user designates the root folder of that hierarchy as the project's working folder. Then the application can automatically create an application folder for each of the child folders in the hierarchy. The child folder becomes the application folder's working folder.

If child application folders are created at the time the project is created, then:

- The application folders' working folders were part of an existing hierarchy on the project creator's workstation.
- Their names are the same as the names of their working folders, but they can be changed later.
- Their working folders remain hierarchically connected to the root folder's working folder. That is, if you change the path to the root folder's working folder, you also change the path to this folder (unless you manually set an absolute path for these working folders). In other words, the application stores a relative path to each child folder.

One of the most important properties to notice about your folder is its working folder. You will need to know where on your workstation the application will copy file revisions that you check out so that you locate those revisions as needed for modifications. A number of other operations can be performed on folders, such as moving a folder or changing its branching behavior.

A working folder is a property of the folder and represents the actual location on your workstation where StarTeam saves files that you check-out. Despite the fact that these are both called folders, the working folder and the folder are not identical. Their differentiating characteristics include:

- The path to the working folder can be totally different from the path within the application to the application folder.
- An application folder is an object controlled from within the application. The data associated with this folder is stored in the database that stores all the project data.
- A working folder is an object controlled by your operating system. It stores files that are checked out from the application.

A project, its root view, and the root folder of the root view all have the same working folder. For additional views, each view and its root folder have the same working folder.

The working folder for the view/root folder always has an absolute path (starting with the drive letter and specifically naming the folders at subsequent levels until you reach the working folder itself).

If you look at the properties for the root folder, you will see that the working folder is the same. However, it is displayed in the **Complete Working Folder Path** display box instead of the **Default** field. Since you can only change the working folder at the view level, all of the fields for the root folder’s working folder are always disabled.

For the child folders that were created at the same time as the project, the application stores the path to each working folder as a relative path.

Folders and Views

You can add new folders and Not-in-View (NIV) folders to views. A NIV folder is a folder on you local disk that does not map to a folder in the StarTeam repository. A NIV folder is displayed as a white folder with a black, dotted border. NIV folders (as with NIV files) do not necessarily need to be added to a view, but you may choose to do so if you just created it and want it to be part of the view. However, if the folder is NIV because someone else deleted it from the view, you may need to delete it from your working folder.

If you add a new folder to a view, its working folder can be any of the following:

- Any folder on your workstation specified by you.
- A non-existing working folder specified by you and created by the application on your workstation. If the existing folder has child folders, one or more of them can also be added to the view.
- A child of the parent application folder's working folder. If you do not specify a working folder, the application appends the new folder's name to its parent's complete working folder path.

Note: If the parent folder's working folder path length exceeds the operating system's maximum working folder path length of 254 characters (including (\) backslashes), the application does not allow you to create the new working folder. Also, you cannot add a folder to a view if the parent folder is read-only. The newly added folder assumes the parent folder's behavior, with a few exceptions. For example, the child folder might have the **Branch On Change** check box disabled because it makes no sense for this folder to branch.

New Folders

You can easily add folders to a project view. When a new folder is added:

- The working folder for the new application folder does not have to belong to the same hierarchy as the other application folders’ working folders. However, if it uses the same drive letter as the root folder’s working folder, its path is stored as a relative path based on the path to the working folder of its parent folder in the hierarchy.
- Its name can be different from the name of its working folder.

- If the new working folder has child folders, a folder can be created for each of the children. Essentially, the newly added folder becomes the root of a new branch of folders. The application folders created for the child folders take the names of their working folders—at least initially. The working folders retain their relationship to the working folder that is the root of their hierarchy (that is, the working folder for the newly added folder). If you change the path to the newly added folder's working folder, you also change the path to these working folders (unless you manually set an absolute path for these working folders).

StarTeam indicates folders on disk that do not map to a StarTeam folder with a **Not-In-View** icon. This indicates that you do not have the folder in the project view.

Existing Folders

You can add more folders to a view by moving them or sharing them from other views on the same server configuration. When a folder is moved or shared, it either keeps its absolute path or its relative path and is applied to its new parent folder. When a moved folder's path is relative, it usually ends up with a different working folder than it previously had. When a shared folder's path is relative, the shared folder has a different working folder in each location.

Note: The application does not allow you to create a working folder if a shared or moved folder's new working folder path exceeds the operating system's maximum working folder path length of 254 characters (including (\) backslashes).

Both the current view and the view from which the folders are moved or shared must use the same server configuration—and, therefore the same database and repository.

Files

To place a file under version control, it must be added to a folder in a StarTeam project view, which stores a copy of the file in the StarTeam repository. After the file has been added to StarTeam, you and other members of your team can check it out, revise it, and check in new revisions, while StarTeam maintains information on all revisions of the file. Note that all check-ins in StarTeam are atomic.

When checking out a file revision, you should verify that you have the *tip* or latest version of the file. Doing this ensures that the file you see contains the latest changes. If you intend to modify the file, you should check it out with an exclusive lock, to indicate to others that you are working on it.

When you check a file in, StarTeam records the file changes as a new revision. As part of the check-in process, you can remove the lock, notifying others that the file is available, or maintain the lock, showing that you intend to continue working on the file. If two team members change the same text file simultaneously or if one member changes an outdated file, StarTeam contains a merge option that allows the file changes to be combined so that no work is lost. In such cases, StarTeam assigns a *Merge* status to the file.

Note: The SDK, StarTeam Server, and most clients support files larger than 4 GB. If you plan on taking advantage of large file support, you should upgrade all users to the current StarTeam version. Large file sizes are not compatible with older StarTeam versions.

Files Under Version Control

If a file resides in the working folder of an application folder, you can add that file to the application folder. This operation places that file under version control. A copy of the working file becomes the first revision of that file stored in the repository. If the working file is deleted later, the data is not lost because a copy exists in the repository. The application creates a new revision of this file in the repository every time you check the file in.

Every time you check a file revision out, its contents are copied to a working folder. Checking out a revision also ensures that you have the tip or a specific revision to work on. For example, you may need a team member's most recent changes to a file, or you may have deleted the working file from your hard drive and now need another copy.

The application enables you to label the tip revisions of every item within a view. For example, when the project reaches a particular milestone (such as beta), you might give the view's items a label, called a view label. Then you can configure the view to return to the way it was at the time the label was applied, check out revisions as a group using that label, create a new view based on the label, or assign the label to a promotion state.

The application also provides revision or version labels. You can label one or more revisions as you check them in or by applying the label to each of the revisions using the **Labels** command on the File menu. StarTeam makes it easy to check out those files as a group using the label. A file revision can have any number of labels. However, no two revisions of the same file in the same view can have the same label.

Recommendations for Working with Files Under Version Control

Here are some recommendations about using files under version control:

- To let other team members know that you intend to make changes to a file, change the lock status to *exclusive* as part of the check-out procedure.
- As part of the check-in process, you can notify others both that you are finished making your changes to the file and that it is available for them to check out by removing the lock status.
- If you intend to continue making changes to the file but still want to check it in for backup purposes, keep the file locked.
- If two team members change the same text file simultaneously or if one member changes an outdated file, you can use the merge option to combine the changes in these files so no work is lost. In such cases, the application gives the file a `Merge` status.
- To prevent yourself from changing a file that you have not locked, select the **Mark Unlocked Working Files Read-only** personal option. Then, if you check out a file that you have not locked, the working copy becomes read-only.

Change Requests

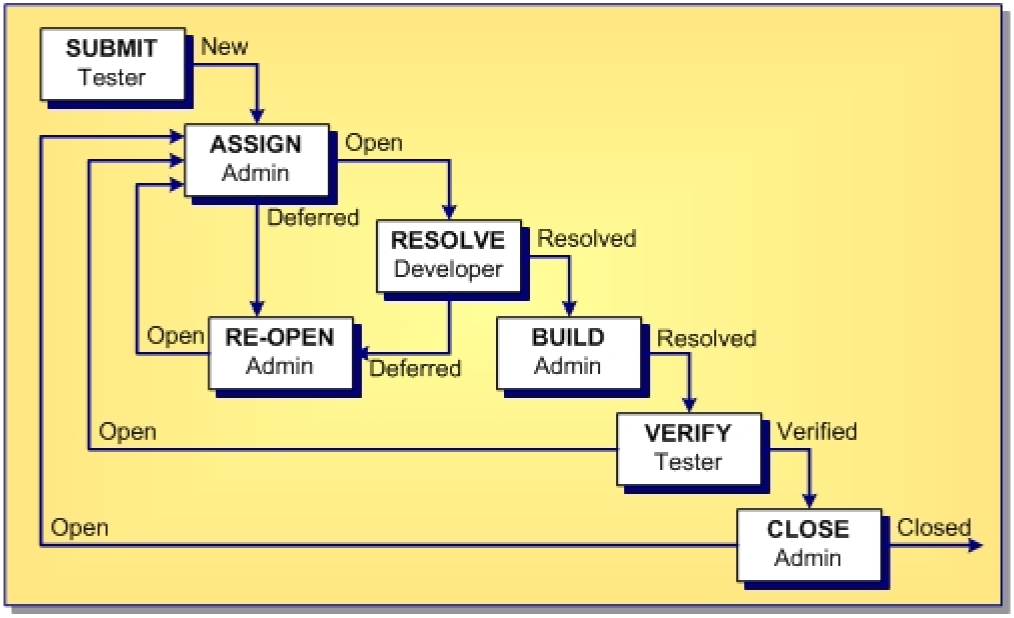
The *change request* component provides a defect tracking system that allows you to record defects in products, projects, or services and suggest possible enhancements. A change request is a request to change something within the scope of a project. For example, you might suggest a product enhancement or request a fix for an error or problem. To use the change request tracking system effectively, you need to understand the model on which it is based.

The change request component allows you to:

- Attach change requests to any folder. In the application, change requests can be attached to any project folder or shared among folders or other views in the same server configuration. You can also link a change request to any other item, such as a file. In many other defect tracking systems, a change request can be associated only with a project, even though it requires modification of a particular file.
- Save time when updating change requests. When you check in a file or group of files, you can indicate the change requests that are fixed by the files being checked in. This feature saves the time required to change the status of each change request separately.
- Make only appropriate status changes. When you create a change request, the status options are `New`, `Open`, `Deferred` or a resolution. The resolutions are `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, and `Is Duplicate`. After resolution, a change request can only be verified or reopened. After verification, a change request can only be closed or reopened.
- Benefit from automatic changes based on the status of the change request. The application automatically changes the person responsible to coincide with the current status of the change request. When a change request is resolved, the responsibility for the change request automatically reverts to the person who entered the change request, who is usually the best person to verify its resolution. When a change request is reopened after being resolved, the responsibility is automatically set to the user who resolved it. If desired, you can override these automatic changes and make another person responsible.
- Base change requests on the build in which the change request is resolved. When a change request receives a **Fixed** or **Documented** status, the value of its **Addressed In Build** field becomes **Next Build**. When that build label is created, the application replaces **Next Build** with the name of the build label, letting testers know the build to use when verifying change requests.

Note: This help system explains how to use the standard property dialog to create and edit change requests. Depending on how your team has set up the application, you may see a different dialog called an alternate property editor (APE). Even if you use the standard property dialog for change requests, your company or team leader may implement change request guidelines that differ from those discussed in this help system.

Change Request Tracking System Model



The above diagram and steps show the change request tracking process. The boxes represent the steps taken from the time that the change request is submitted until the time it is closed. Each box indicates an action and the team member most likely to be responsible for performing this action. The arrows show the status of the change request at the time of each step.

The change request tracking system consists of the following steps:

Step 1 A team member creates a new change request that does either of the following:

- Summarizes a problem with the product and lists the steps taken to reproduce the problem.
- Suggests an enhancement to the product.

This change request has a status of `New`.

Step 2 Another person, such as an administrator or team leader, decides whether to fix the problem or add the suggested enhancement to the product. This person can:

- Set the status of the change request to `Open`, and assign a team member to resolve it.
- Set the status of the change request to `Deferred` because it is worthwhile but will not be done at this time.
- Set the status of the change request to `Is Duplicate` because this is not the first time it has been submitted. If desired, a link can be created between a change request and the original submission so that you can track the change request along with the original submission.
- Set the status of the change request to `As Designed` because the product is supposed to work this way, meaning there is no defect.

Change requests with an `Open` status go to step 3.

Step 3 The person assigned to resolving the change request changes the status of the change request to `In Progress`. Later on, after this person finishes examining the change request, he or she changes the status to one of the following:

- `Fixed`
- `Documented`
- `Cannot Reproduce`

Step 4 Next, a team member (usually a tester or quality assurance engineer) verifies the change request. For example, a test case may be developed to determine if the problem is really fixed, documented or not reproducible and changes the status to one of the following verified statuses:

- `Verified As Designed`
- `Verified Cannot Reproduce`
- `Verified Documented`
- `Verified Fixed`
- `Verified Is Duplicate`

Step 5 Finally, another team member changes the status to `Closed`. This person may then perform activities related to closing the change request, such as retesting the change request before closing it or adding it to a report to be included in the next release of the product.

In most of the above steps, the change request can be reopened and reprocessed.

Built-in Workflow for Change Requests

StarTeam has a built-in workflow for change requests that automatically sets many of the values associated with change requests. This built-in workflow determines these settings based on the setting of the **Status** field for the change request.

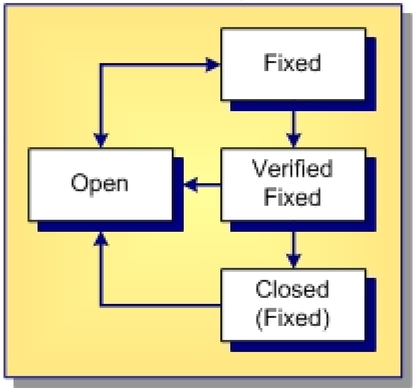
You cannot add additional settings to the **Status** field. However, you can rename them to better suit preferences set by your organization. For example, your organization may prefer to change the name of the value `New` to `New Change Request`.

When you alter the status of a change request, the built-in workflow automatically selects the appropriate properties associated with the change in status.

After selecting `New`, `Open`, or `In Progress`, six new statuses display in the **Status** list. These statuses, which are associated with the status you selected, are:

- `Deferred`
- `Cannot Reproduce`
- `As Designed`
- `Fixed`
- `Documented`
- `Is Duplicate`

Lifecycle for Change Requests



The above diagram shows the lifecycle for a change request with an initial status of `Open`. The status was then set to `Fixed`. After this setting, the built-in workflow added an additional status field of `Verified Fixed`. Finally, the change request was closed, meaning its status was set to `Closed (Fixed)`.

The diagram also shows that a change request can be reopened at any stage in its lifecycle because the arrows leading from each of the three fixed statuses can lead back to the **Open** status at any time.

Summary of the Change Request Automatic Workflow

The following summarizes the steps used in processing change requests as explained in this topic. It includes the automatic workflow changes the application makes to change requests based on their statuses.

Submit

Anyone (usually a tester or quality assurance engineer) can submit a change request.

Process: Select the **Change Request** tab. Then select **New** from the **Change Request** or context menu.

A change request has the following default properties (which you can change if necessary).

Status: `New`

Severity: `Low`

Priority: `Not prioritized`

Type: `Defect`

Platform: `All`

Last Build Tested: `Current build label`

Entered by: `Person currently logged on to the application`

Many other fields are initially blank. Some team leaders prefer to have all change requests submitted at the root folder. They use drag-and-drop to move the change requests to the appropriate child folders.

Assign*

Process: The team leader finds all new change requests and does one of the following:

- Opens the change request and assigns it to a developer, help writer, or other appropriate team member.
- Defers the change request until a later date, perhaps the next release of the product.
- Specifies that the change request is `As Designed` and not to be fixed. If the change request status is `Open`, no automatic changes occur. If the change request status is `Deferred` or `As Designed`, then `Addressed in Build` is disabled and the responsibility is assigned to the user who created the change request.

Resolve

Process: Users find the `Open` or `In Progress` change requests assigned to them, and do one of the following for each request:

- Resolve the problem in the system and update the properties of the change request. (The statuses that indicate that a change request has been resolved are `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, or `Is Duplicate`.)
- Defer the change request until a later date, perhaps the next release of the product. Your team leader may prefer that you do not defer change requests.
- If the change request status is one of the possible resolution statuses, then **Addressed in Build** becomes `Next Build` for `Fixed` and `Documented` statuses. It becomes disabled for other statuses. By default, the responsibility is assigned to the person who submitted the change request, who is expected to verify the resolution.

- If the change request status is `Deferred`, then **Addressed in Build** is disabled and the responsibility is assigned, by default, to the user who created the change request.

Build

Who builds the project? The project view may have a formal or informal build process. However, at some point, all the files, and so on currently in the view receive that build label. It is usually applied to the source code files, and so on that were compiled (and may need to be changed) rather than to the executable files that result from the build.

Effect on change requests: For any resolved change request that has `Next Build` as the setting for its **Addressed In Build** property, `Next Build` is replaced with the next build label that is created.

Note: If a new build label is based on a past configuration (rather than the current configuration), it has no effect on the `Addressed In Build` property.

If a change request has not branched in its current location, `Next Build` may be replaced with a build label from another view. For example, suppose you create a branching child view or share a folder from one view to another. Suppose that `Next Build` is the value of some change request's `Addressed In Build` property and that change request has not branched. When a build label is created in the source view, `Next Build` is replaced with the name of that build label, regardless of the location.

Verify*

The person who submitted the change request (usually a tester or quality assurance engineer) verifies a resolution.

Process: Install the build in which the resolution is to be verified and determine whether the change request has been resolved correctly. Do one of the following:

- Verify the change request, marking it as `Verified Cannot Reproduce`, `Verified As Designed`, `Verified Fixed`, `Verified` Documented, or `Verified Is Duplicate`.
- Reopen the change request and update the setting for **Last Build Tested**.
- If the change request status is `Verified`, no automatic changes occur.
- If the change request status is `Open`, **Addressed in Build** is blank. If the change request has changed from resolved to **Open**, the user who changed the status to `Fixed` or `Documented` becomes responsible.

Close*

Usually the team leader closes the change request.

Process: The team leader does one of the following:

- Reviews and closes the verified change request.
- Reopens the change request.
- If the change request status is `Closed`, then no automatic changes occur.
- If the change request status is`Open`, then **Addressed in Build** is blank. If the change request has changed from resolved to `Open`, the user who changed the status to `Fixed` or `Documented` becomes responsible.
- If the status of a change request changes from `Verified` to `Open`, the user who changed the status to `Fixed` or `Documented` becomes responsible and **Addressed in Build** is blank.

*Changes in status can result in automatic changes to other properties.

Requirements

Requirements are supported for the Enterprise Advantage license and display in the **Requirement** tab of the upper pane in the clients. With the requirement component, you can create requirements within the application and show the dependencies among them. For example, if one requirement must be fulfilled before a second requirement can be fulfilled, the first can be made a child of the second. If your company enforces process rules, the requirements you establish can also be used to drive the development process. Administrators and other authorized users can publish requirements from Caliber to StarTeam using Publisher to StarTeam, which is delivered with Caliber.

Requirement Characteristics

The requirements in the upper pane have the following characteristics:

- They are attached to the folder selected from the folder hierarchy.
- They match the filter selected from the **Filter** list.
- They match the depth specified by **All Descendants**.

  Tip: You can click the button on the toolbar or click **Requirements** > **All Descendants**.

  Note: You can click the **All descendants** button on the **Requirements** view toolbar.

Note: Icons display to the left of a requirement in the upper pane to indicate its status and whether you have read the latest revision.

How Requirements Can Help

By using a requirements-driven development processes, companies can prevent consuming, costly misunderstandings and shorten time to market. To accomplish this, you can use the StarTeam built-in requirement component as your basic tool, or publish complex requirements to StarTeam from Caliber. Using requirements enables business analysts, managers, developers, QA staff, and others to:

- Organize business, user, and functional requirements in a hierarchical format.
- Indicate the dependencies among requirements.
- See all layers of requirements at all times.
- Prioritize requirements by importance.
- Identify the impact of changes to requirements.
- Use requirements to estimate work.
- Identify the person creating the requirement.
- Notify those who will be responsible for fulfilling the requirements.
- Track the requirement life-cycle from submitted to completed or rejected.
- Provide requirements with a context by linking them to files, change requests, and topics.

Tasks

The *task* component allows the creation of task lists and work assignments. As a standalone, the task component is very useful for managing a project. It allows team members to indicate who should do what and when, see current task status, estimate hours required to complete a task, record hours spent completing the task, and compare estimated to actual times. Because the application contains both a version control system and a change request system, it also allows tasks to be linked to the files and product defects or suggestions with which they are associated.

The task component can be used independently or interoperate with data from Microsoft Project. It can display tasks in a tree format, which clearly shows the relationship between tasks and subtasks, or in a list format, which allows tasks to be sorted, grouped, or queried, or specific fields to be selected for display. To improve efficiency, each task displays icons that identify its status, priority, milestone, and need for attention. For information about interoperating with Microsoft Project, see the *StarTeam Microsoft Project Integration User's Guide*.

With the StarTeam task component, you can create an individual task or a summary task that has a set of subtasks. It is recommended that you plan tasks before entering them because:

- A task that has even one subtask cannot have work records added to it, although work records can be added to subtasks. The application assumes that the name of the task indicates a goal, perhaps a milestone, that will be reached when the subtasks are completed.
- After a work record has been added to a task, you cannot create subtasks for it.

Note: Regardless of whether work records can be added to a task, you should assign the responsibility for its completion to a specific team member. If work records can be added to a task, you should also estimate how long the task should take.

Topics

Topics are threaded conversations, that is, a series of messages that indicate how the messages are related. Each series of messages forms a tree with the initial message at its root. The topic component provides threaded conversations that you can place in specific project folders and link to specific project items. For example, you can link a topic to the change requests and file revisions that result from the topic discussion.

The upper pane of the client consists of topics and a series of responses to each topic. A series of topic trees are eventually formed, each of which consists of a root topic and its responses. The topic tree resembles a conversation that may go on among several people. In the client, this is called a threaded conversation because a topic and its responses are threaded together, starting with the root topic. By reading each response in a thread, one after the other, and the responses to those responses, you can see how the discussion has evolved. A number of other operations can be performed on topics or responses such as moving or sharing them.

Historical Value of Topics

Topics can raise general questions about the project or start very specific discussions about issues, such as feature implementation. While the responses can lead to resolution of these issues, the historical value of these conversations to the project can be even more significant. Future team members can:

- Reassess decisions more capably.
- Avoid retrying solutions that were previously found faulty.
- Understand why a particular solution to a problem became necessary and, therefore, not replace that solution with one that does not meet all the necessary criteria.

How Topics Can Help

Any type of threaded messaging improves teamwork on product development. However, because StarTeam has tightly integrated components, it enables team members to:

- Search topics and responses for specific words or phrases.
- Sort topics and responses.
- Filter topics and responses.
- View relationships between topics and their responses.
- Move and share topics (from the tree format).
- Link topics directly to folders or other items, such as change requests.
- Ask questions and quickly receive input while working on a file.
- Attach notes to a topic explaining why a particular method was used.
- Point out aspects of the project that may need to change in a later release.

Links: Internal and External

A link is a connection between two folders, two items, or a folder and an item on the same server, or on two different servers (called *External Links*). You can view links by using the **Link** tab in the lower pane of the StarTeam Cross-Platform Client.Creating links can be quite useful. For example, linking a file to a change request allows you to mark it as fixed when you check in the edited file. By linking files to the requirements document that the files fulfill, you can easily refer to or update the document.

In addition, linking files to change requests enables you to mark the change requests as fixed when you check in the corresponding files. In turn, if you link each set of files to the requirements document that the files fulfill, you can easily refer to or update the document.

A link does not provide a connection to a single share (or reference), but to all related shares and branches of an item. Links are not affected by any item operations, such as branching, moving, sharing, and so on. By default, a link connects the tip revisions of the linked pair.

Links can either be pinned or floating:

Pinned You can lock the link to the tip revision. The context menu in the link tab enables you to pin links to the source or target items or both.

Floating You allow the link source or target to change from tip revision to tip revision as new revisions are created. The context menu in the link tab enables you to float links to the source or target items or both.

Links, as with all other items, have context menus in their tabbed panes which allow you access to more information about the item.

Labels

In version control, the term `label` corresponds to the act of attaching a view or revision label (name) to one or more folders/items. StarTeam enables you to create two types of labels:

View labels Are automatically and immediately attached to all folders and items in a view at the time you create the view label. View labels have multiple purposes, but you primarily use them to place a *time stamp* on the entire contents of the view and as *build labels*. When you roll back the view to that label, you see everything that existed at that point in time—unless the label has been adjusted. You can create a view label for a specific point in time or as a copy of another existing view label. Unless the view label is frozen, you can adjust it to include or exclude some folders and items by attaching or detaching view labels. You can also move a view label from one revision to another.

Revision labels Are not attached automatically to any item in the view. Instead, they are used to designate a set of folders or items within a view. For example, you might want to label a group of files that should be checked in and out together. After you create a revision label, you attach specific items, building it up to reflect a specific set that is typically a small subset of the view. StarTeam can automatically attach new file revisions to a revision label at check-in time if you like.

About Labels

You can attach a label to any type of StarTeam item, including folders, files, requirements, change requests, tasks, topics, and audit entries. Any item can have more than one label. However, no two revisions of the same item can have the same label at the same time.

Every label is unique within its view. That is, no view label can have the same name as any other view label, no revision label can have the same name as any other revision label, and no view label and revision label can have the same name. Each view has its own set of labels. This also means that every label has a *name* that must be unique from other labels belonging to the same view. Each label also has a *description* that helps users understand the purpose of the label.

You can manually attach or detach both view labels and revision labels to or from a folder or item. In addition, you can use either type of label to identify a file when it is checked out. When you check a file in, you can attach and create a revision label for that file or attach an existing revision label.

You can select any type of item by its label. For example, you can select all files with a particular revision label and roll them back to that label, making the revision with that label the tip revision. Then you can compare your working files to the rolled-back revisions.

You can set access rights for labels at the view level or at the folder or item level. You must grant the rights to create labels, edit their properties, and delete them at the view level. However, you can grant the right to move a label (also called *adjust a label*) at the folder or item level.

A label can be frozen, which means no new artifacts can be attached to it, and attached artifacts cannot be detached nor reattached at a different revision. Conversely, non-frozen labels can have all of these modifications. Since many organizations depend on the immutability of frozen labels, a specific security permission is required to freeze or unfreeze a label.

StarTeam actually attaches specific *item revisions* to a label, each of which infers a specific artifact revision. Since each item specifies a parent folder, the artifact is attached in a specific folder. This means that if you move an item, you need to reattach it to any label for which you want to reflect the artifact in its new folder. If you don't reattach a moved item to an existing label, it will continue to be attached to the label in its old folder (which may be what you want).

Even if frozen, both view and revision labels can be cloned. That is, you can create a new label starting out identical to an existing label and then adjust the revisions attached to it. A common practice is to clone a previous label, attach only new file revisions that were created to fix a bug, and use the new label to identify the file revisions for a new build candidate or release candidate.

Time Stamps and Build Labels

Using a view label as a *time stamp*, you can roll a view back to see everything in the view as it was at the time the label was attached. For example, to see if a particular file was in the beta version of a product, you can roll back the view to the beta label.

You may also use a view label as a *build label*, which allows the QA team to immediately determine what build to test for a fix to any given change request. To use a view label for this purpose:

- It must be designated as a build label.
- It must be created while the **Addressed in build** property for the change request contains the value `Next Build`.

When StarTeam creates the label, each change request with Next Build as its **Addressed in build** property will be reset to the build label.

To create a view label, you must select the current configuration of the view. Historical configurations are read only, and adding a label is considered a change. However, if a label already exists for a prior configuration, you can adjust its name, files and folders can be added to it or detached from it, and so on. You can also move a view label from one revision to another.

For example, suppose your administrator creates a view label before each build, giving that label to the tip revision of each file in the view, and then checks out all the files with that label for the build. If the tip revision of one file does not change for a few weeks (or longer), it can acquire several view labels, while a file that changes frequently may have several revisions with no view labels and other revisions with only one view label.

When you detach a view label from a folder, StarTeam automatically detaches the label from everything in the subtree for which the folder is the root. If you roll back a view to a specific view label and a folder does not have that label, you cannot see the children of that folder and their contents anyway.

You can only create a view label at the view level and only while the configuration is current. However, you can create a view label for the current configuration or for a time in the past. In either of these two cases, StarTeam attaches the new label to the tip revision of each folder, file, change request, task, or topic that belonged to the view at the specified time.

You can also create a view label as a copy of an existing view label or as a copy of the view label currently attached to a promotional state. In these two cases, StarTeam attaches the new label to exactly the same items and revisions as the existing view label.

You can check file revisions out using this label or roll back the view to this label and see all the items associated with that label. For example, if you create the view label *Build 100* as you make Build 100 of your product from a view, all the files in the view will have the label *Build 100*.

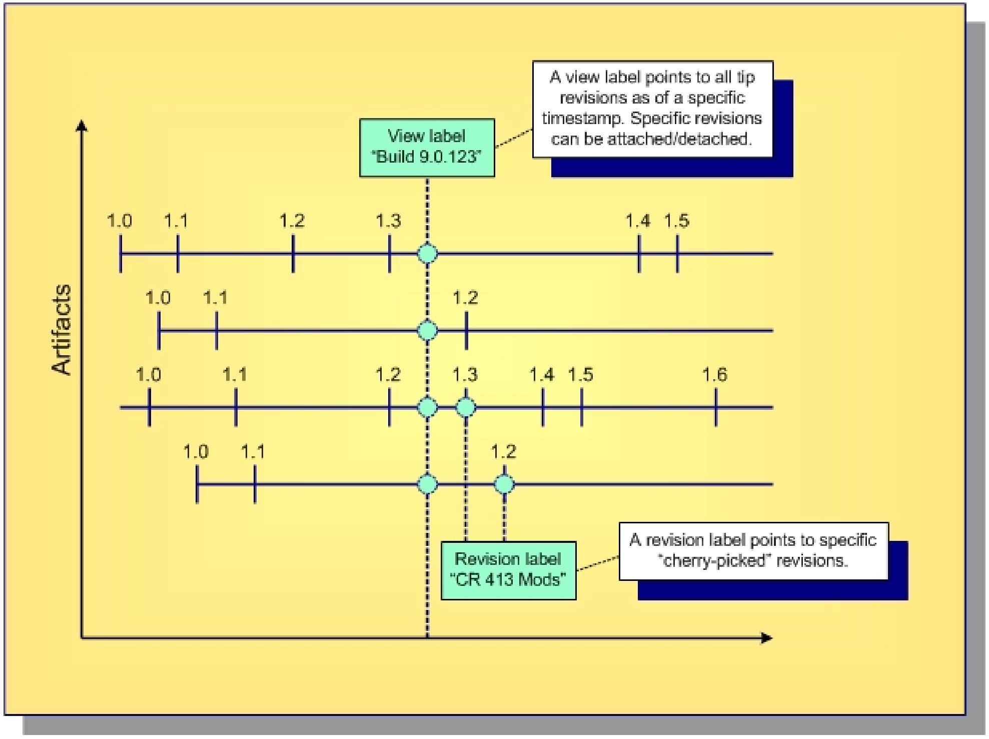
If some items should not be included, you can detach the label from those items individually. For example, if some files should not have that label, select the files then select **Labels** > **Detach** from the **File** menu or context menu to detach that label. If the files that should not be included all belong to the same folder and are the only files in that folder, use the **Labels** command on the **Folder** menu. For example, if the help files were not checked in until after the view label was attached, you can move that label from the previous revisions of the help files to the newly checked-in help files.

Label Access Rights

You can set access rights that apply to labels at the view level and at the folder/item levels. You set the access rights that allow a user or group to create labels, edit their properties, and delete them at the view level. For example, if you can create a label, you can set its initial properties. However, if you do not have the right to edit label properties, you cannot later freeze or unfreeze that label.

You can attach labels to individual folders or items, detach from them, or move from one of their revisions to another. The access right to move a label is named **Adjust a label**. You can grant or deny these rights at the folder or item level.

Example View and Revision Labels



A view label points to all tip revisions as of a specific timestamp. Specific revisions can be attached/detached.

A revision label points to specific "cherry-picked" revisions.

Note: Not all items in a view have to be attached to a label. Conversely, an item can be attached to any number of view and revision labels as you like, but only one revision of an item can be attached to any specific label.

Branching, Merging and Dot Notation

This topic explores the concepts of branching, merging and dot notation as they relate to StarTeam.

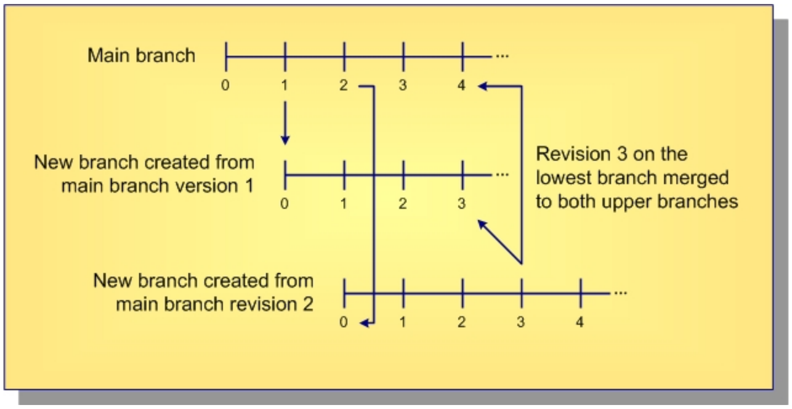
Branching Allows an artifact’s version stream to be forked. Each fork can be independently modified, receiving its own versions.

Merging Propagating a change on one branch to another branch.

Dot Notation A dotted decimal notation assigned to artifact revisions to indicate both the branch on which the revision resides and the relative version number of the revision within the branch (for example: 1.4 or 1.2.1.5).

Branching

There’s a big difference between *copying* an artifact and *branching* an artifact. Although copying allows each copy to be modified independently, StarTeam does not know that copied artifacts are related to each other in the repository. In contrast, with branching, StarTeam retains special knowledge about an artifact’s branches. This information supports things such as intelligent revision comparison and *three-way merging*, which is discussed below.


Main branch
0
1
2
3
4
...
New branch created from main branch version 1
0
1
2
3
...
Revision 3 on the lowest branch merged to both upper branches
New branch created from main branch revision 2
0
1
2
3
4
...

When a new artifact is added to the repository, a *main branch* is started and new revisions are added to it. At any time, a parallel *child* branch can be started. In the example above, one child branch is created from main branch version 1, and another child branch is created from main branch version 2. Within a branch, the version number starts over (at zero in this example). New revisions are applied to a specific branch, incrementing the version number on that branch but not affecting other branches. Branching is needed to support parallel development on files. However, there are advantages to allowing non-file artifacts to branch as well. For example, if a defect artifact can be branched, the two branches can be used to track fixes to the same defect that exist in different releases.

Merging

Inevitably, a change on one branch will need to be propagated to another branch. In the diagram above, version 3 of the artifact on the lowest branch is applied to both of the parent branches. However, you can’t just copy a revision from one branch to another branch—this could wipe out changes that are specific to the target branch. Instead, what you want to do is merge the changes from the source to target branch.

Note: *Overwriting* the target artifact with the source revision instead of merging it is sometimes the desired effect, for example with binary files.

More specifically, StarTeam stores synchronization information that allows *three-way merging*. The three parts of a merge operation are:

1. The *source revision* containing changes to be propagated.
2. The *target revision* that will be modified.
3. The last source revision common to the source and target branches, known as the *common ancestor*.

For files, merging is done by passing these three file revisions to the **StarTeamFile Compare/Merge** tool. The tool compares both the source revision and target revision to the common ancestor revision and determines two important things:

1. What changes appear in the source revision only that should be propagated to the target revision.
2. What changes appear in the target that may conflict with changes in the source revision.

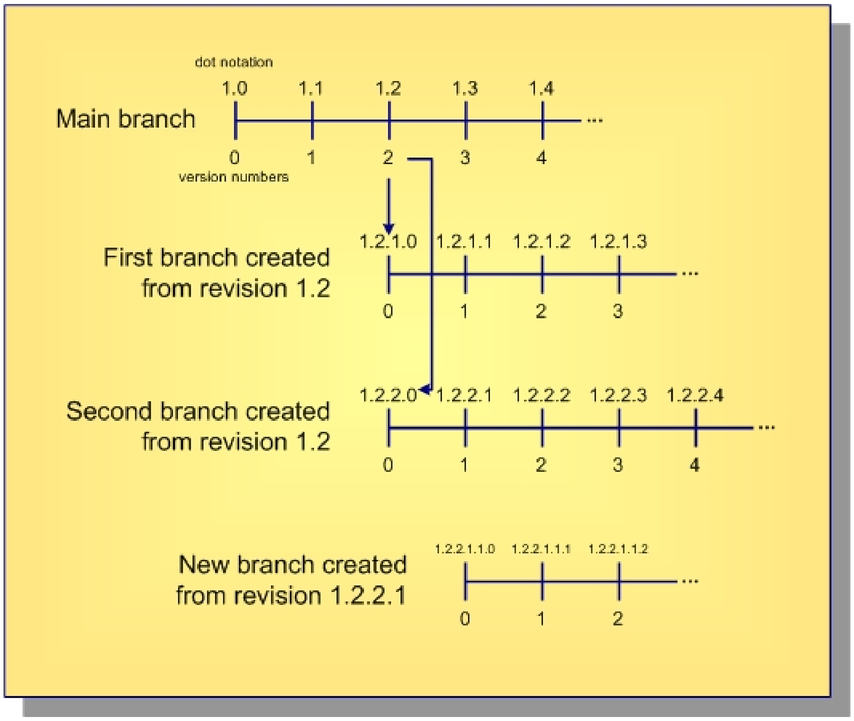
In many cases, merging detects no conflicts, so the **StarTeamFile Compare/Merge** tool automatically propagates the source changes to the target revision. When conflicts are detected (and sometimes even when none are), the **StarTeamFile Compare/Merge** tool displays the differences to a user who can review the differences, resolve conflicts, and approve the final result. The **StarTeamFile Compare/Merge** tool then creates a result file reflecting the target file updated with changes. StarTeam adds the result file to the target revision's branch, creating a new revision.

In the diagram above, revision 3 on the lower child branch was merged to both of the upper branches. When it was merged to the main branch revision 3, it created main branch revision 4, which contains the merge results. (The arrow points to the revision that was created as a result of the merge.) For this merge, the common ancestor between the two branches is main branch revision 2: it is the most recent revision that both branches had in common. When the lower child revision 3 is merged with upper child branch revision 2, upper child branch revision 3 was created. For this merge, the main branch revision 1 is the common ancestor.

For files, there is more to merge than contents: files have other properties such as `name` and `description`. In order to propagate a name change, for example, merging a file requires merging these properties as well. Non-file artifacts that branch also require merging in order to propagate changes.

Dot Notation

In addition to a version number, StarTeam assigns each revision a dotted-decimal value called a *dot notation*. Whereas the version number is unique within a revision branch, the dot notation value is unique within the entire *revision tree*. An example is shown below.



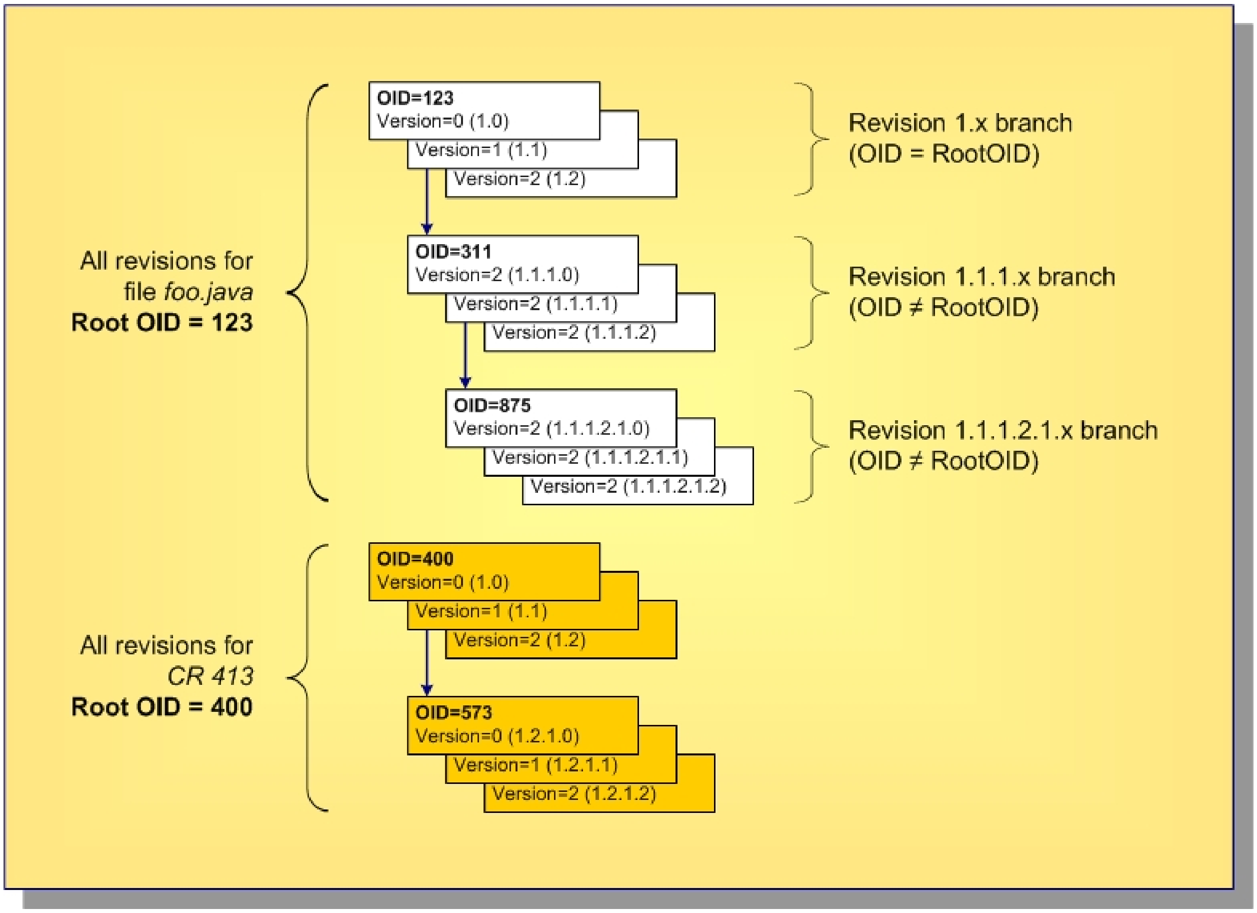
As shown, revisions on the artifact's *main branch* use the single dot notation pair `1.n`, where `1` indicates that it is the initial (first) branch and n is the same as the version number. When the artifact is branched from the main branch, revisions on the child branch use the dot notation `1.m.1.n`, where `m` is the main branch version number from which the branch was created and *n* is the version number on the new branch.

Note: An artifact can branch more than once from same point: in the example above, branches `1.2.1.n` and `1.2.2.n` were both created from main branch revision `1.2`. The second 2 in `1.2. 2 .n` tells you that this is the second branch from revision `1.2`. Branch `1.2.2.1.1.n` has three pairs of numbers, telling you that it is a third-level branch, created from parent revision `1.2.2.1`.

Artifacts that can't branch (tasks, topics, and requirements) are always on the main branch, so their dot notation is always `1.n`.

Object IDs and Root Object IDs

All revisions in the same revision tree have the same *root object ID* (root OID). All revisions that belong to the same branch have the same object ID (OID). Furthermore, for all revisions on the main branch, the object ID and root object ID are the same. This is illustrated below.



In this example, the file `foo.java` started with OID and root `OID 123`, and the corresponding 1.n branch has revisions up to `1.2`. At revision `1.1`, it branched to form the `1.1.1.n` branch, which uses the new `OID 311`. Revision `1.1.1.2` was branched to form the `1.1.1.2.1.n` branch with `OID 875`. But all revisions in the entire branch tree have the original root `OID 123`. Each revision holds the properties specific to it: name, description, contents, etc.

Also shown is a change request (`CR 413`) that began with OID and root OID equal to 400. At revision `1.2`, it branched to form branch `1.2.1.n` with `OID 573`.

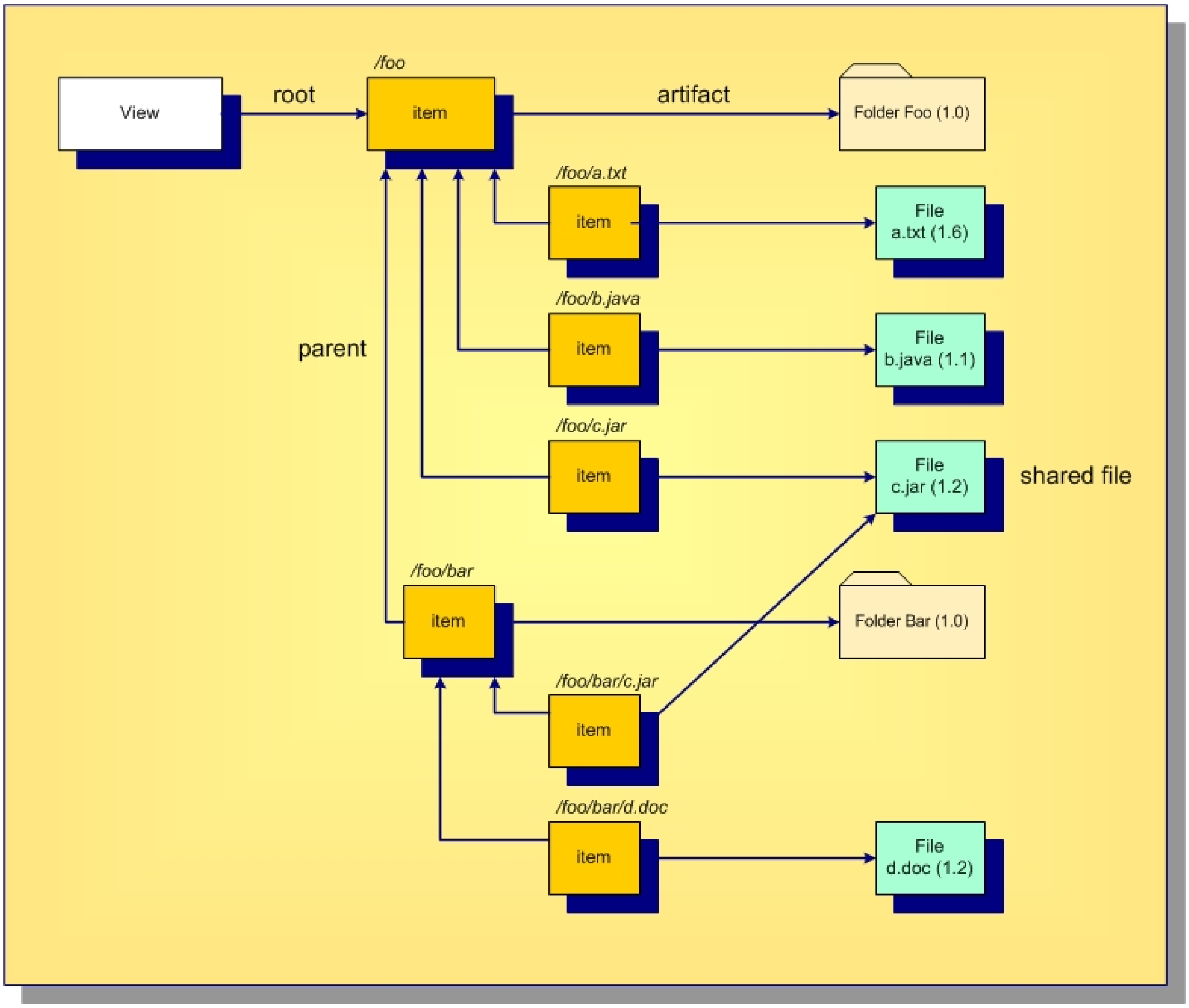
Now, consider a *folder* artifact. Each revision holds properties such as `name`, `description`, and `exclude spec` (file patterns to ignore within working folders). What's really different about StarTeam folder artifacts is that they do not have a property that represents their contents.

Sharing and Cheap Copies

Over time, you will have a lot of artifacts, especially files, and some files will have a lot of branches. Consider the effect of containers: if you have a lot of teams, software components, and releases, you will need a lot of independent projects, subprojects, or other containers to support parallel development and separate maintenance. Often the same files will be needed in each of these containers. How do you get the files you need to each of these containers? Forcing every file to branch in order to get a unique branch in every possible container could be a lot of branching, which is expensive.

StarTeam systems addresses this problem with a technique known as *cheap copies*. This involves creating references to files in a new container. Similar to UNIX links, this happens without actually copying the files themselves (that is, their content or their history). Unlike UNIX links, however, the first time a file is modified via a new reference, it is branched. For this reason, cheap copies are also referred to as “copy on write” sharing. Cheap copies support efficient branching with large projects.

In StarTeam, the folder hierarchy and the contents of each folder are specific to each view. Artifacts can belong to (or more properly be exposed through) any number of views and projects. Items are objects that select specific artifacts, connect them to a specific view, and organize them into a hierarchy. The diagram below shows how this works.



Every view has a *root item*, which always points to a folder artifact. In this example, the root folder name is `foo`. We can make any artifact in the repository belong to this folder by creating an item that points to the artifact we want and the root item as the `parent`. In this example, the files `a.txt`, `b.java`, and `c.jar` and the folder `bar` are all child elements of the root folder`foo`. As you can see, the concept of `path name` is formed by concatenating the names referenced by the item structure. In this view, there is a file whose path name is `/foo/bar/d.doc` because we can get to this artifact via the item path: folder `foo` to folder `bar` to file `d.doc`. If we want to change the folder in which `d.doc` appears, we change the `parent` of its associated item, the artifact itself is not modified. Notice that two items reference the file `c.jar`. This means that this file is contained in two different folders. We say that the file is `shared` in two places. This is analogous to UNIX links that reference the same file, causing it to appear in multiple directories. Sharing allows any artifact to be shared in multiple places. Since artifacts are “heavy” (they contain all the properties) and items are “light”, this is how “cheap copies” are made: we just create items pointing to existing artifacts.

Setting Access Rights

1. Select the object or item in the client for which rights will be set:
   - To set project access rights, the project must be open to any view.
   - To set view access rights, the view must be open.
   - To set folder access rights, select the folder from the folder hierarchy in the left pane.
   - To set component access rights for public filters or queries, (for example, the change request component) bring the component tab into focus in the upper pane.
   - To set individual file, change request, requirement, topic, or task item access rights, select the item from the upper pane.
2. Select **Access Rights** from the appropriate menu or context menu as follows:
   - To set project-level access rights, select **Project** > **Access Rights** .
   - To set view-level access rights, select **View** > **Access Rights** .
   - To set folder-level access rights, right-click the folder on the **Folder** tab and choose **Advanced** > **Access Rights** .
   - To set component-level access rights for public filters and queries, select the component type, (for example the **Change Request** menu) **<Component Type>** > **Advanced** > **Component Access Rights** .
   - To set item-level access rights, select the item from the upper pane, and select **<Component Type>** > **Advanced** > **Item Access Rights** .
3. Click **Add**. The **Assign Access Rights To** dialog box opens.
4. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
5. Select **Grant** and click **OK**.

   ⚠ **Caution:** Never select **Deny** unless you are creating an exception. Deny records must be created before grant records.
6. Select/clear the appropriate check boxes. Selecting or clearing the check box for a category, such as **Generic object rights** for a project, selects or clears all the access right check boxes for that category. The category check box has only two states. When it is cleared, the access right check boxes for that category are either all cleared or mixed: some selected and some cleared.

   ⚠ **Caution:** Clicking **Delete** removes the selected user or group from the **User and Groups** list in the **Access Rights** dialog box. The selected user or group loses any previously set access rights to the Server.
7. Click **OK**.

General Access Rights Rules

Keep these general rules in mind when granting access rights:

- Access rights can be overridden by the fact that a user is the object’s owner. Usually, the owner is the person who created the object.
- Access rights can be overridden by privileges given to a group that includes the user. These privileges are set per group from the Server. By default, the **Administrators** group has full privileges (rights to do anything and everything).
- Access rights should be set at the highest possible level.
- The client checks for access rights from the lowest level (the item level) to the highest level (the project level).

- If one grant record is created for a node, a grant record for that node should be created for every group that requires access to the project at that level. The **Administrators** group should have a grant record for each node, so that, if privileges are ignored, administrators can still change access rights.
- If access rights are set for any user or group for a node, all users or groups without a grant record for that node will be denied all access rights at that level for that node.
- Every view within a project has the same project-level access rights.
- When you derive a child view from an existing view, the new view has no view-level access rights. However, folders and items in the child view that existed in the parent view retain the same folder-level or item-level access rights that they had in the parent view. Changing these access rights in either the parent or the child view also changes them in the other view because you are changing the rights on the same object. If the folders or items in either the parent view or the child view branch, they can have different access rights, because they are different objects.
- Folders that are moved or shared from one view to another retain any access rights assigned to them at the folder level in the new view. However, if they branch, they lose their folder-level access rights.
- Items that are moved or shared from one view to another retain any access rights assigned to them at the item level to the new view. However, if they branch, they lose their item-level access rights.
- Avoid setting item-level access rights.
- Avoid creating deny records. But if you deny rights, follow both of these rules: a) never allow any node on an **Access Rights** dialog to have only deny rights records, and b) verify that deny rights records for a node precede any grant rights records for the node.

Group Privileges and Access Rights

When users log onto a server configuration, they are identified individually by their user names and as members of the groups to which they belong. The application stores this information as an access token for each user. As users perform operations on application objects (projects, views, folders, and items), the application examines these tokens and the access rights for the objects on which the users are performing the operations. The application checks access rights in layers. The right to access an object begins with the **System Policy** dialog, which can be accessed from the Server Administration tool.

Unless group privileges are being ignored, these privileges also override and take precedence over rights configured elsewhere. Privileges are group properties set on the **Privileges** tab of the **Group Properties** dialog in the client. A user is granted the same privileges as those of the group to which he or she belongs. If the user belongs to two groups, and one is granted certain privileges while the other is denied the same privileges, the user is granted the privileges. The **Membership** tab of the **My Account** dialog displays the logged-on user's group membership information.

After consideration of group privileges, the application checks the access rights granted for specific objects. Settings on the **Access Rights** dialogs for projects, views, folders, and individual items grant or deny users or groups the ability to perform operations at those levels. It is important to remember that if access rights are granted to any user or group at a given level in an **Access Rights** dialog, users or groups who are not granted access rights at that level are effectively denied all rights.

Ultimately, if a user can see an object and is not stopped from performing an operation by a deny record, the user can do anything that a grant record allows, whether as an individual user or as a member of any group. The only exception has to do with privileges.

Denying Access Rights

For a given node at a given level, grant records are examined until one gives a user or group permission to perform an operation or until all the grant records have been examined without finding one that gives permission. If membership in one group does not allow a user to perform an operation but membership in a second group does, the user can perform the operation. However, if a deny record for that node forbids the user from performing an operation, the user cannot perform that operation. The application disregards any grant records for the same node that allowed the user to perform the operation.

Deny Record Considerations

Deny records are rarely used. However, they do allow you to create exceptions to the current access rights. Keep these considerations in mind:

- Deny records must precede grant records. The reason is that if the application finds a grant record that allows a user to perform an operation before it finds a deny record for the user, it stops looking at records for that node at that level. Thus, it allows the user to perform the operation.
- Creating a grant record with no check boxes selected is not the same as creating a deny record with all the check boxes selected, although both stop users and groups from performing the same operations.
- Group privileges can override either grant or deny records.

Deny Records for Projects

Before deleting a project from StarTeam, you may consider hiding it from the users. Creating one deny record at the project level for the `All Users` group (or for another umbrella group of users accessing the project) denies those users the access rights to see the project. It is essentially hidden from view and cannot be accessed for the group that has been denied.

Creating a Deny Record to Handle Access Right Exceptions

Suppose that you have a group called `Testers` that has complete access to the files in the `QA` view, a view that contains folders full of test files. A newly hired member of the `Testers` group, `New Tester`, has not yet been trained to update the tests, and so on. Although `New Tester` is a member of the `Testers` group, you do not want this user to perform certain operations on these files for a couple of weeks. You could remove `New Tester` from the `Testers` group temporarily, but the application also allows you to give `New Tester` all the rights of the `Testers` group with a few exceptions. To list the exceptions, you create a deny record.

1. Click **Add**. The **Assign Access Rights To** dialog box opens.
2. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
3. Select **Deny** and click **OK** to return to the **Access Rights** dialog box.

   Caution: Never select **Deny** to create an exception to a group unless that group is already specifically granted access for this same node. In this example, the Testers group must have access for this node.
4. Select/clear the appropriate check boxes. Selecting or clearing the check box for a category, such as **Generic object rights** for a project, selects or clears all the access right check boxes for that category. The category check box has only two states. When it is cleared, the access right check boxes for that category are either all cleared or mixed: some selected and some cleared.

   Caution: Clicking **Delete** removes the selected user or group from the **User and Groups** list in the **Access Rights** dialog box. The selected user or group loses any previously set access rights to the Server.
5. Click **Move Up** to move the deny record to the top of the **Users and groups** list in the **Access Rights** dialog box.

   Tip: All deny records must precede all grant records in the **Users and groups** list. Otherwise, the exception will not occur. For example, if the application finds the grant record for Testers before it finds the deny record for New Tester, the rights the user has as a member of the Testers group will apply.
6. Click **OK**.

Note: Depending on the privileges of the `Testers` group, `New Tester` may be able to perform these operations anyway. Also, if a deny record is the only record for a node, anyone not specifically granted access rights for that node has no access to that type of object at that level. When the application finds a node for the correct type of object that has even one record, it does not check higher levels for access rights.

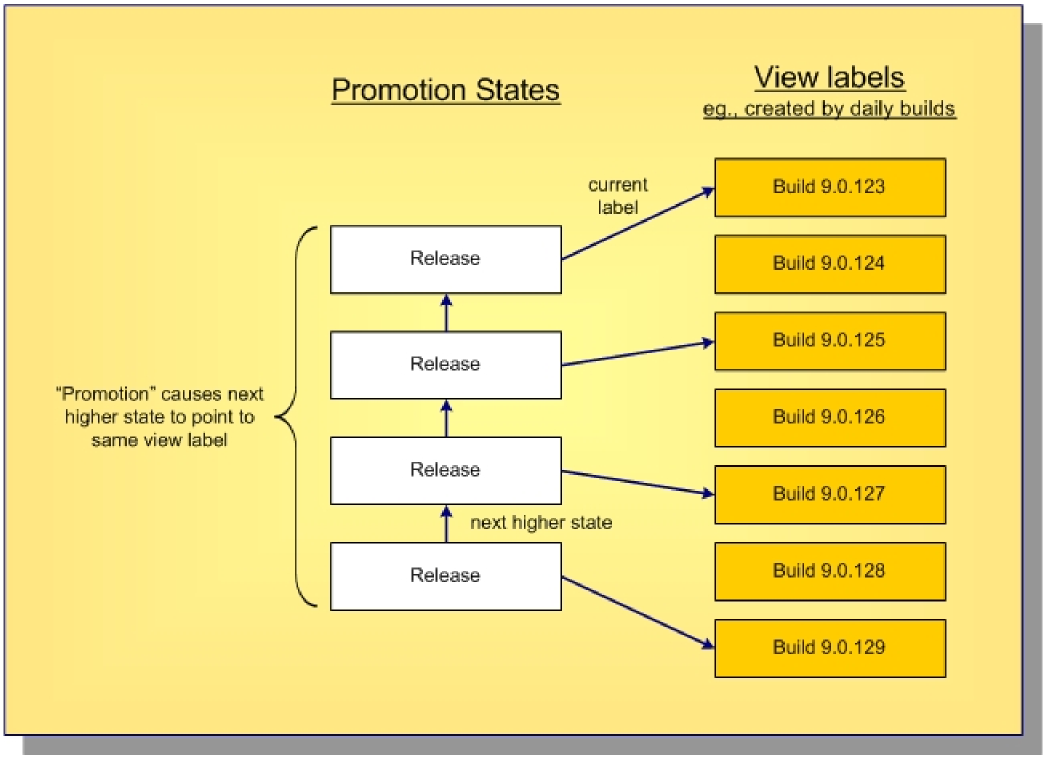
Promotion States

Promotion states provide an *intra view* change management facility. Promotion states are built on top of view labels, providing an ordered set of states through which items can be promoted within a view. Promotion states are generally used to move the entire view (or most items within it) through a series of steps based on passage of specific verification tests.

After defining a set of promotion states and what view label each state is initially mapped to, you then periodically create new view labels to represent specific view states (such as daily build candidates). Depending on your process, you then typically map the lowest-level state to the new view label and launch the first verification test. After the tests for that state complete, you "promote" it, causing the next higher state to be mapped to that view label. (Multiple states often point to the same view label.) When the tests for the final or "top most" promotion state passes, the view is ready for release, deployment, or whatever your process calls for.

Promotion states allow you to create build scripts, unit test scripts, deployment scripts, and so forth that operate on a specific promotion state without having to be modified to know about new view label names.

An example set of promotion states is shown below.



Audit Log

The audit log is a record of events that happen to your assets. Click the **Audit** tab to display audit log entries for the selected view.

Audit Log Events

Events are actions performed on an owner. For example, a file can be checked in or removed from version control. Such events are recorded in the audit log. Most items can be:

- Added

- Branched
- Comment Edited
- Created
- Deleted
- Locked
- Lock Broken
- Modified
- Moved From
- Moved to
- Shared
- Unlocked
- Converted
- Edited
- Item Overwritten (as foreign archive files become native files)
- Vault
- Created
- Modified
- Deleted
- Frozen
- Unfrozen
- Attached
- Moved
- Detached
- Modified

Audit Fields

This section lists all the audit fields in alphabetical order.

Class Name 1

Values: text

Internal Identifier: `Class Name 1` (contains spaces)

The name of the class of items, such as Label, Promotion State, Folder, File, Change Request, Topic, Task, or Trace.

Class Name 2

Values: text

Internal Identifier: `Class Name 2` (contains spaces)

The name of the class of items, such as Folder, File, Change Request, Label, Topic, Task, or Trace.

Class Name 3

Values: text

Internal Identifier: `Class Name 3` (contains spaces)

The name of the class of items, such as Folder, File, Change Request, Label, Topic, Task, or Trace.

Created By

Values: list of users, `<None>`

Internal Identifier: `CreatedUserID`

Always empty because the audit entry is created by the system.

Created Time

Values: date/time

Internal Identifier: `CreatedTime`

The time at which this entry was created.

Deleted By

Values: list of users, `<None>`

Internal Identifier: `DeletedUserID`

The name of the user who deleted an audit entry. Because deleted entries do not appear in the list, this information is unavailable to users.

Deleted Time

Values: date/time

Internal Identifier: `DeletedTime`

The time at which an audit entry was deleted. Because deleted entries do not appear in the list, this information is unavailable to users.

Event

Values: `Added, Branched, Comment Edited, Created, Deleted, Edited, Item Overwritten, Label Attached, Label Created, Label Deleted, Label Detached, Label Frozen, Label Modified, Label Moved, Label Unfrozen, Lock Broken, Locked, Modified, Moved From, Moved To, Promotion Model Modified, Promotion State Modified, Shared, Unlocked, Vault Converted`

Internal Identifier: `EventID`

The name of the operation being recorded.

Folder

Values: text

Internal Identifier: `Folder`

The name of the folder that stores the audit entry.

Folder Path

Values: text

Internal Identifier: `Folder Path` (contains spaces)

The path to the folder that stores the audit entry.

Folder VMID (Advanced)

Values: number

Internal Identifier: `FolderVMID`

The ID assigned to the folder that stores the item.

Item 1

Values: text

Internal Identifier: `Item 1` (contains spaces)

Indicates what class `1` item received the audited operation. This can be the name of a file or task, the number of a change request or requirement, or the title of a topic.

Item 1 Info

Values: text

Internal Identifier: `Info`

Provides the revision number in dot notation for the class `1` item, if it is revisionable. For example, a label can be a class `1` item and it does not have revisions.

Item 2

Values: text

Internal Identifier: `Item 2` (contains spaces)

Indicates what class 2 item received the audited operation. For example, if a label was attached to a file, the class 1 item is the label and the class 2 item is the file.

Item 2 Info

Values: text

Internal Identifier: `Info2`

Provides the revision number in dot notation for the class 2 item, if it is revisionable. For example, a label can be a class 2 item and it does not have revisions**.**

Item 3

Values: text

Internal Identifier: `Item 3` (contains spaces)

Indicates what class 3 item received the audited operation. For example, if a label was moved from one revision to a file to another, the class 1 item is the label, the class 2 item is the revision of the file that was initially , and the class 3 item is the final revision of the file.

Item 3 Info

Values: text

Internal Identifier: `Info3`

Provides the revision number in dot notation for the class 3 item, if it is revisionable. For example, a label can be a class 1 item and it does not have revisions.

Modified By

Values: list of users, `<None>`

Internal Identifier: `ModifiedUserID`

Does not apply to audit entries.

Modified Time

Values: date/time

Internal Identifier: `ModifiedTime`

Does not apply to audit entries.

Object ID

Values: number

Internal Identifier: `ID`

Each audit entry is assigned an object ID when it is added to a view.

Project

Values: list of projects in this server configuration, `<None>`

Internal Identifier: `ProjectID`

The name of the project in which an audit entry was recorded.

Target 1 Class ID (Advanced)

Values: number

Internal Identifier: `Target 1 Class ID` (contains spaces)

The ID number assigned to class 1 items or a `-1` if there is no ID.

Target 1 Object ID (Advanced)

Values: number

Internal Identifier: `Target 1 Object ID` (contains spaces)

The object ID for the class 1 item that received the audited operation or a `-1` if there is no ID.

Target 1 Revision Time

Values: date/time

Internal Identifier: `Target 1 Revision Time` (contains spaces)

The time at which the last revision was made to the class 1 item that received the audit operation.

Target 2 Class ID (Advanced)

Values: number

Internal Identifier: `Target 2 Class ID` (contains spaces)

The ID number assigned to class 2 items or a `-1` if there is no ID.

Target 2 Object ID (Advanced)

Values: number

Internal Identifier: `Target 2 Object ID` (contains spaces)

The object ID for the class 2 item that received the audited operation or a `-1` if there is no ID.

Target 2 Revision Time

Values: number

Internal Identifier: `Target 2 Revision Time` (contains spaces)

The time at which the last revision was made to the class 2 item that received the audit operation.

Target 3 Class ID (Advanced)

Values: number

Internal Identifier: `Target 3 Class ID` (contains spaces)

The ID number assigned to class 3 items or a `-1` if there is no ID.

Target 3 Object ID (Advanced)

Values: number

Internal Identifier: `Target 3 Object ID` (contains spaces)

The object ID for the class 3 item that received the audited operation or a `-1` if there is no ID.

Target 3 Revision Time

Values: date/time

Internal Identifier: `Target 3 Revision Time` (contains spaces)

The time at which the last revision was made to the class 3 item that received the audit operation.

Transaction ID (Advanced)

Values: number

Internal Identifier: `TransactionID`

Uniquely identifies the database transaction that contained the update represented by the audit record. (A database transaction can contain multiple updates.) Note that audit records created before the database was upgraded to a StarTeam release that records a **Transaction ID** will have a **Transaction ID** of `-1`.

User

Values: list of users, `<None>`

Internal Identifier: `UserID`

The name of the user who performed the recorded operation.

View

Values: list of views, `<None>`

Internal Identifier: `ViewID`

The name of the view in which an audit entry was recorded.

Table of Common Operations

The following table lists the generic operations that can be performed with each component.

Operation	File	Change Request	Requirement	Task	Topic	Folder	Audit
Moving	Yes	Yes	No	Yes, except when using Microsoft Project Integration.	Yes	Yes	No
Drag an item to a new location	No	No	No	No	No	Moving a folder moves its contents, child folders, and their contents.	No
Creating shortcuts to items	Yes	Yes	Yes	Yes	Yes	Yes	No
Copying items to a third-party application via a URL	Yes	Yes	Yes	Yes	Yes	Yes	No
Sharing `Ctrl+drag` item to a new location	Yes	Yes	Yes	Yes	Yes	Yes	No
Branching behavior	Yes	Yes	No	No	No	Yes	No
Configuring to or freezing at a point in the past	Yes	Yes	Yes	Yes	Yes	Yes	No
Locking	Yes	Yes	Yes	Yes	Yes	No	No
Comparing properties of two items of the same type	Yes	Yes	Yes	Yes	Yes	No	No
Comparing properties of two revisions	Yes	Yes	Yes	Yes	Yes	Yes	No
Review revision history	Yes	Yes	Yes	Yes	Yes	Yes	No
Viewing revision properties	Yes	Yes	Yes	Yes	Yes	Yes	No

Operation	File	Change Request	Requirement	Task	Topic	Folder	Audit
Editing revision comments	Yes	Yes	Yes	Yes	Yes	Yes	No
Merging revisions	Yes, using Visual Merge.	No, except as a part of merging views.	No, except as a part of merging views.	No, except as a part of merging views.	No, except as a part of merging views.	No, except as a part of merging views, which is often done by an administrator	No
Finding based on field content	Yes	Yes	Yes	Yes	Yes	No	Yes
Selecting by query	Yes	Yes	Yes	Yes	Yes	No	Yes
Selecting by label	Yes	Yes	Yes	Yes	Yes	No	No
Label revisions	Yes	Yes	Yes	Yes	Yes	Yes	No
Viewing references	Yes	Yes	Yes	Yes	Yes	Yes	No
Linking to folders and items	Yes	Yes	Yes	Yes	Yes	Yes	No
Printing a default report for selected items	Yes	Yes	Yes	Yes	Yes	No	No
Sending items as email	No	Yes	Yes	Yes	Yes	No	Yes
Receiving email notification about changes (when notification is enabled by administrator)	No	Yes. Changes in responsibility only.	Yes. All changes in items for which you are responsible.	Yes. All changes in items for which you are responsible.	Yes. All changes in items for which you are responsible.	No	No
Controlling system tray notification	No	Yes	Yes	Yes	Yes	No	No
Marking items as read/unread	No	Yes. You can also mark trees as read/unread.	Yes. You can also mark trees as read/ unread.	Yes. You can also mark trees as read/unread.	Yes. You can also mark trees as read/unread.	No	No
Flagging items	Yes	Yes	Yes	Yes	Yes	No	No
Deleting	Yes	Yes	Yes	Yes	Yes	Yes	No

Operation	File	Change Request	Requirement	Task	Topic	Folder	Audit
Setting access rights. Normally performed by Administrator	Yes	Yes	Yes	Yes	Yes	Yes	No
Creating reports	Yes	Yes	Yes	Yes	Yes	No	Yes
Creating charts	Yes	Yes	Yes	Yes	Yes	No	Yes

Testing and Reporting Overview

As a quality assurance engineer, you may perform some or all of the following actions:

- Enter and resolve change requests.
- Update the status of change requests.
- Generate charts and reports based on change request data.
- Extract server data using StarTeam Datamart.
- Create custom queries and filters for displaying change request data.
- Version test plans.
- Delete change requests.

Change Requests

The *change request* component provides a defect tracking system that allows you to record defects in products, projects, or services and suggest possible enhancements. A change request is a request to change something within the scope of a project. For example, you might suggest a product enhancement or request a fix for an error or problem. To use the change request tracking system effectively, you need to understand the model on which it is based.

The change request component allows you to:

- Attach change requests to any folder. In the application, change requests can be attached to any project folder or shared among folders or other views in the same server configuration. You can also link a change request to any other item, such as a file. In many other defect tracking systems, a change request can be associated only with a project, even though it requires modification of a particular file.
- Save time when updating change requests. When you check in a file or group of files, you can indicate the change requests that are fixed by the files being checked in. This feature saves the time required to change the status of each change request separately.
- Make only appropriate status changes. When you create a change request, the status options are `New`, `Open`, `Deferred` or a resolution. The resolutions are `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, and `Is Duplicate`. After resolution, a change request can only be verified or reopened. After verification, a change request can only be closed or reopened.
- Benefit from automatic changes based on the status of the change request. The application automatically changes the person responsible to coincide with the current status of the change request. When a change request is resolved, the responsibility for the change request automatically reverts to the person who entered the change request, who is usually the best person to verify its resolution. When a change request is reopened after being resolved, the responsibility is automatically set to the user who resolved it. If desired, you can override these automatic changes and make another person responsible.

- Base change requests on the build in which the change request is resolved. When a change request receives a **Fixed** or **Documented** status, the value of its **Addressed In Build** field becomes **Next Build**. When that build label is created, the application replaces **Next Build** with the name of the build label, letting testers know the build to use when verifying change requests.

Note: This help system explains how to use the standard property dialog to create and edit change requests. Depending on how your team has set up the application, you may see a different dialog called an alternate property editor (APE). Even if you use the standard property dialog for change requests, your company or team leader may implement change request guidelines that differ from those discussed in this help system.

Queries

You can use a query to limit the items that display. Each query is performed on all items in the StarTeam folder and component you have selected. The fields included in the query do not have to display. Once a query has been created, it can be used in every project in the same server configuration.

StarTeam queries have the following attributes:

- A unique name that easily identifies the query. Query names are not case-sensitive.
- Public or private status. Anyone with appropriate access rights can use public queries, while private queries are available only to your user ID. Once a query has been saved with a specific status, its status cannot be changed. However, you can copy a query and change the state of the new query.
- A logical expression appropriate for items of a particular type. These expressions include one or more conditions. A condition consists of a field (not necessarily a current column header), a relational operator, and a value to be compared to the value of the field. For example, a condition used to locate change requests might be: `Responsibility Equals Rhonda Thurman`. More complex queries include two or more conditions bound together by logical operators: `AND`, `OR`, and `NOT`. For example, to locate all the change requests for which `Rhonda Thurman` is responsible that also have a `high` severity, use: `Responsibility Equals Rhonda Thurman AND Severity Equals High`.

Note: If you are creating a complex query, and the first logical operator in your query should be `OR`, select the `AND` logical operator in the query tree. Then click the **AND->OR->NOT** button. This changes an `AND` to an `OR`. Similarly, one more click changes the `OR` to a `NOT`. Keep toggling the button until the operator that appears is the one you want to use. It is best to use the condition or logical operation that will result in the fewest matches as the first condition or logical operation.

Classic Reports

StarTeam offers a wide variety of *Classic* reports that are pre-formatted in HTML. However, you can modify the HTML formatting or the columns used in the report by modifying the correct template for the report. You can also use StarTeam Datamart to extract data from StarTeam Server and place the data into a relational database where reporting tools (Crystal Reports and Business Objects) can access it.

Report generation is affected by sorting, grouping, and selecting items in the file, change request, topic, or task list of the view window. Before you generate a report, arrange the data in the upper pane of the client.

This topic provides some examples of why you might generate reports and information about customizing report templates. Reports are generated for a variety of reasons. The following examples describe three possible scenarios for generating a report using change requests and a brief walkthrough on how to generate them.

Although these examples apply only to change requests, you can create additional reports for other items. You can also export data for use in spreadsheets and other applications.

Manager wants a report of all CRs resolved across a project

1. The manager selects the root folder from the folder hierarchy, and selects the **Change Request** tab. The list of change requests display in the upper pane.

2. The manager selects the `Status=Resolved` filter from the **Status** list.
3. The **All Descendants** (either from the toolbar or the **Change Request** menu) button is activated.
4. Finally, he or she selects **Change Request** > **Reports** to generate a **Change Request Summary Report**.

Team leader wants to see CRs based on responsibility

1. The team leader selects the project folder that corresponds to his or her team from the folder hierarchy, and selects the **Change Request** tab. The list of change requests display in the upper pane.
2. The team leader selects the `<Show All>` filter from the **Status** list.
3. The **All Descendants** button is activated.
4. The team leader clicks the header of the **Responsibility** column to sort the change requests based on the responsible team member.
5. Finally, the team leader selects **Change Request** > **Reports** from the main menu to generate a **Change Request Detail Report**.

Developers want to see CRs for which they are responsible

1. The individual programmers select their project folders from the folder hierarchy, and select the **Change Request** tab. The list of change requests display in the upper pane.
2. The programmer applies a query (`Responsibility Equals <username>`) to view only his or her assigned change requests.
3. The **All Descendants** button is activated.
4. Finally, the programmer selects **Change Request** > **Reports** to generate a **Change Request Detail Report**.

Report Templates

StarTeam allows you to customize report templates. The templates are located in the folder you designated during installation. For example, if you used the default installation path for the Cross-Platform Client on a Windows platform, the `Reports` folder is `C:\Program Files\Micro Focus\StarTeam Cross-Platform Client <version>\Reports`. Be aware that different clients and different releases of just one client will probably have different installation folders. You may need to put templates in more than one location. All the templates are in HTML format.

You can open and edit the report templates in any text editor, Microsoft's Developer Studio, or HTML tool. A simple and easy method of creating and editing templates is to use Microsoft Word, which includes automated HTML file generation.

The `Reports` folder includes a series of templates for each type of report. Each template provides the formatting information needed to create a part of the report. For example, the **Change Request Default** report uses the following templates:

- `ChangeDefault.Title`
- `ChangeDefault.GrpInfo`
- `ChangeDefault.Group1`
- `ChangeDefault.EndReport`

`xDefault.Title`
Processed first and only once. It uses the title you provide in the **Reports** dialog as the **Report Title**.

`xDefault.GrpInfo`
Processed once for each group. Although you can sort data by clicking column headers, this does not result in groups for a report. You must use the **Sort and Group** feature to arrange groups.

`xDefault.Group1` Repeated for each item, in this case, each change request, in the report. It creates a record for the item and separates it from the records for other items with a horizontal line.

`xDefault.EndReport` Processed only once. It totals the number of items in the report.

Charts

StarTeam offers a wide variety of charts. The Cross-Platform Client allows you to create simple, distribution, correlation and time-series charts of your data. The charts you can create depend upon the component tab menu option that you have selected. For example, the charts available for files differ from those available for change requests.

Charts are created from the data displayed (maximum of 60 fields) in the upper pane. To select the data to be used for a chart, you can show or hide all descendants of a folder, sort and group items, and run queries and filters.

You can use charts in a number of different ways. For example, you can use charts to track the number of closed and newly-opened change requests during a time period of a product development cycle.

You can also filter out data in the upper pane of the Cross-Platform Client to display only the data that you want to include in your chart. You can select specific items from the filtered data to include in your chart.

Filters

A filter is a named arrangement of data that consists of a set of fields (used as column headers), sorting and grouping information, and (usually) a query. Once a filter has been created, it can be used in every project that has the same server configuration.

Filter names are not case sensitive. For example, if you have a filter named `recent CRs`, you cannot create a filter named `Recent CRs`, as StarTeam considers the two filters to be identical. In the **Filters** list, filters display in alphanumeric order, but you can control the order in which they appear by carefully naming or renaming them.

If you set up a filter and do a **Send to** in the client, only the fields displayed by the filter are sent to the recipient.

You can filter data in the upper pane in several different ways:

- By applying an existing filter.
- By arranging the data (changing displayed fields, sorting and grouping the files, and so on) and applying a query. You can then use this arrangement as the basis for a new filter.
- By creating a new filter from scratch.

Note: Only private queries can be used in private filters, and only public queries can be used in public filters. Therefore, you cannot copy a filter and change the status of the new filter unless the filter does not include a query.

Fields

StarTeam provides both common and advanced fields that you can display as columns in the upper pane or use in queries along with relational operators that you can use with those fields to define conditions. To see an advanced field in the **Show Fields**, **Sort and Group**, or **Query** dialog boxes, you must select the **Show Advanced Fields** check box.

Note: Client-calculated fields cannot be used in custom email notifications or with the StarTeam Notification Agent.

Data Export with Datamart

StarTeam Datamart is a complementary product to StarTeam Server. StarTeam Datamart uses the StarTeam SDK to communicate with StarTeam Server to create a reporting database that you can use with popular third party reporting applications such as Crystal Reports and Business Objects (reporting applications are not included with StarTeam Datamart).

StarTeam Datamart extracts data from a StarTeam server and places the data into a relational database, where reporting tools can access it. The data is stored in meaningful text instead of IDs.

StarTeam Datamart can extract information from every project, every view in each project, every folder in each view, and every item in each folder, as well as labels, links, and history for each item. You can restrict the extracting to a particular project and view, and also restrict to extract only for certain tables.

A sample Business Objects universe and sample reports in both Business Objects and Crystal Reports formats are included. These samples can be used as a basis for creating custom user defined reports.

Datamart Extractor	Extracts the data from a StarTeam Server and puts the data into a relational database. For StarTeam administrator use.
Datamart Synchronizer	Opens the universe and synchronizes the data to reflect any field changes. For StarTeam administrator use.

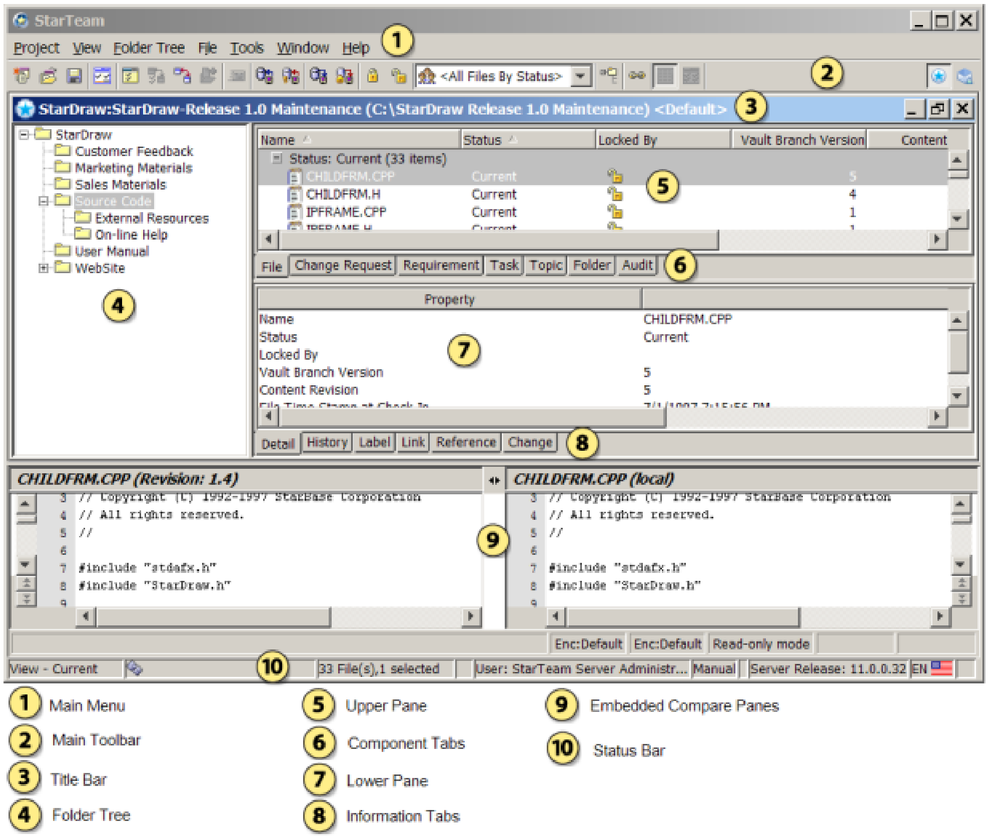
Note: The Synchronizer and Viewer work with Business Objects 6.1 universes and reports. The Viewer also works with Crystal Reports 10 reports. You can view StarTeam Datamart reports with the Crystal Reports Viewer.

Tour of the UI

This section contains topics describing the StarTeam Cross-Platform Client user interface.

Cross-Platform Client UI

The main window of the Cross-Platform Client is also called the project view window because it shows not only a project but a particular view of the project. That view may be the root (or initial) view with which the project was created. The **Content Perspective** view is the default view that you see when you open the Cross-Platform Client. See the numbered components in the image and refer to the descriptions below.


StarTeam
Project View Folder Tree File Tools Window Help
<All Files By Status>
StarDraw:StarDraw-Release 1.0 Maintenance (C:\StarDraw Release 1.0 Maintenance) <Default>
StarDraw
Customer Feedback
Marketing Materials
Sales Materials
Source Code
External Resources
On-line Help
User Manual
WebSite
File Change Request Requirement Task Topic Folder Audit
Detail History Label Link Reference Change
1
2
3
4
5
6
7
8
9
10

Name	Status	Locked By	Vault Branch Version	Content
Status: Current (33 items)				
CHILDFRM.CPP	Current		5	
CHILDFRM.H	Current		4	
IPFRAME.CPP	Current		1	

Property	
Name	CHILDFRM.CPP
Status	Current
Locked By	
Vault Branch Version	5
Content Revision	5

CHILDFRM.CPP (Revision: 1.4)

```
// Copyright (C) 1992-1997 StarBase Corporation
// All rights reserved.
//

#include "stdafx.h"
#include "StarDraw.h"
```

CHILDFRM.CPP (local)

```
// Copyright (C) 1992-1997 StarBase Corporation
// All rights reserved.
//

#include "stdafx.h"
#include "StarDraw.h"
```

Main Menu

The contents of the main menu vary depending upon the selected component. In this example, the File component has focus, so the **File** menu displays as a selection in the main menu. Possible components include File, Folder, Change Request, Requirement, Task, Topic, and Audit. Menu items that always display in the main menu, regardless of the selected component, include **Project**, **View**, **Folder Tree**, **Tools**, **Window**, and **Help**.

Toolbar

The buttons on the toolbar differ according to the selected component. Frequently used main menu commands have corresponding buttons on the toolbar. Fly-over text displays when you hover your mouse over the toolbar buttons.

At the right of the toolbar are two buttons for switching between two perspective windows of the StarTeam information:

- : The standard **Content Perspective** which is the perspective you are accustomed to using for Cross-Platform Client.
- : The **Change Perspective** which allows you to view and manage all your View Compare/Merge change packages from past VCM sessions.

Title Bar

The title bar at the top of the main window displays the following information:

- Server configuration that contains the currently displayed project view
- Project name
- View name
- Working folder for the view

Folder Tree

The left pane of the main window displays the folder tree or folder hierarchy. These folders have a one-to-many relationship with the contents of the upper pane because each folder can contain different types of items (for example, files and change requests). Using the Folder Tree in the left pane, you can perform the following operations:

Display information using icons

A folder icon precedes each folder name. Clicking this icon displays folder-associated information in the upper pane. The information displayed depends on the folder and the tab selected on the upper pane.

Expand and collapse branches

You can expand or collapse branches of the folder tree. A plus sign identifies a collapsed branch. Clicking it expands the branch. A minus sign identifies an expanded branch. Clicking it collapses the branch. Alternatively, you can double-click folder names to expand or collapse branches of the folder tree.

You can also move through the folder tree using the **Up Arrow**, **Down Arrow**, **Left Arrow**, and **Right Arrow** keys. The **Right Arrow** expands a folder. The **Left Arrow** collapses a folder.

Right-click for context menu

After selecting a folder from the folder tree, right-click to display a context menu that presents the same choices as the **Folder Tree** menu.

Drag items and folders

To move a child folder, you can click on it and drag it from one parent folder to another. You can also drag items from the upper pane to the tree in the left pane for the current view or another view (if both are in the same server configuration). To move items from one folder or view to another, select the items, and then drag them from one location to the other.

Upper Pane

The upper pane of the main window contains a tree or list of items associated with the folder that displays in the folder tree. Although each folder can contain items of different types, the upper pane displays only one type of data at a time. The type of data displayed depends upon the component tab selected.

The contents displayed in the upper pane depend upon the following factors:

- The folder selected from the folder tree in the left pane.
- The component tab selected from the upper pane.
- The filter selected from the **Filter drop-down list**, which displays in the toolbar above the upper pane.
- The **All Descendants** toolbar button, found in the toolbar above the upper pane. The **All Descendants** command determines the depth for which the client displays information. When not selected, the client displays information for the selected folder only. When selected, the client displays information for the selected folder, its children, its children’s children, and so on.

Component Tabs

The upper pane of the main window contains component tabs that represent the components that your license provides. Clicking on one of the component tabs in the upper pane brings that component tab into focus.

StarTeam Enterprise Advantage licenses provide all components:

- **File**
- **Change Request**
- **Requirement**
- **Task**
- **Topic**

- **Folder**
- **Audit**
- **Custom**: optional tab. Appears only if the administrator creates a custom component.

Lower Pane and Information Tabs

The lower pane of the main window displays information about the item selected on the upper pane. The five information tabs on the lower pane show different types of information. All of the tabs apply to all of the upper pane components except the **Audit** component. When you select the **Audit** component, you see only the **Detail** tab.

Detail tab

The **Detail** tab lists the information displayed on the upper pane in a horizontal format. For example, if you select a file from the upper pane, the lower pane will show its name, size, status, and so on, depending on the columns that appear in the upper pane.

You can modify the display format and content of the **Detail** (lower) panes in the StarTeam Cross-Platform Client on a per-workstation basis.

History tab

The **History** tab lists the past revisions for the selected item. The only exception is a file with the `Not In View` status, in which case the tab displays nothing at all.

Label tab

The **Label** tab lists the labels associated with each revision of the selected item. You can use drag-and-drop to move a label from one revision to another.

Link tab

The **Link** tab shows the items to which the selected item is linked. For example, a file might be linked to a folder, another file, or a change request. Items can also be linked externally from one server to another. The Link tab decorates the external link in the View column with a special external link decoration (), and provides a checkbox which you can use to show or not show external links.

Reference tab

The **Reference** tab shows the relationships between the selected item and other folders or items with which it is associated. A folder or item may be associated with more than one project, view, or parent folder in the same server configuration because of sharing or because a child view has been created. Each instance of the original folder or item has a reference.

Change tab

The **Change tab** shows all the actions performed on a process item or workspace (check-in) change package.

Embedded File Compare/Merge

The File Compare/Merge tool is also embedded in the client and when activated, is available for use with the **File**, **History**, and **Link** (if the link is to a file) tabs. You can use this tool in its embedded format to show the differences in the file that you have selected in the upper pane and your local file.

By default, the client does not display the embedded version of File Compare/Merge. You can toggle this feature on and off using the **Tools** > **File Compare** main menu command.

Note: File Compare/Merge also opens in standalone mode. You can open File Compare/Merge using the start menu.

Status Bar

By looking at the **Status** bar at the bottom of the main window, you can determine the following:

- Whether a view is current, rolled back to a label, rolled back to a promotion state, or rolled back to a specific time. If a view is read-only, `R/O` displays before the configuration information. All rolled back views are read-only because you cannot change the past.
- Whether you have selected an active process item. For example, if you have select a change request for your active process item, the status bar displays the change request title along with its associated number.
- The number of items in the upper pane and the number of items selected.
- Whether you have applied a query to the upper pane.
- Name of the logged-on user.
- The status of StarTeamMPX. A lightning bolt icon represents the status of StarTeamMPX. The lightning bolt changes color or shows a small red "disconnected" symbol in the lower left corner to let you know if StarTeamMPX is available, available but disabled, or available and disconnected. The words **Instant**, **Auto**, and **Manual** provide refresh status information.
- The version of StarTeam Server that the client is accessing.
- The locale and default language of the client.
- Whether you have new (or unread) change requests, requirements, tasks, or topics assigned to you.

Project Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New Starts the **New Project Wizard**, and creates a new project.

Open Starts the **Open Project Wizard**, and opens an existing project.

Close Closes the current project.

Open Shortcut Opens the project view associated with the shortcut in the configuration it had when the shortcut was created. The shortcut displays each view in the configuration it had at the time the shortcut was created.

Save Shortcut As Creates a new shortcut to the current project view. Shortcuts are usually saved to your desktop.

Open StarTeam URL Opens StarTeam URL shortcut links to projects, views, folders, not-in-view folders, and items. This enables you to quickly access specific locations in a project.

Open StarTeam Agile When enabled, this option opens StarTeam Agile in a new browser window.

Recent Projects Reopens a recently-opened project view.

Properties Opens the **Project Properties** dialog box for the current project.

Access Rights Opens the **Project Access Rights** dialog box for the current project

Delete Deletes the item from the StarTeam repository.

Connection Properties Opens the **Connection Properties** dialog box for the current project and displays the current server description, address, endpoint, user name, and logon time for the connection.

Log Off Closes all the views on the server that you have logged into during this session.

Exit Exits the application.

View Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New Starts the **New View Wizard**, and creates a new view.

Delete Deletes a view from the current project.

Select View Opens a dialog box displaying the view tree for the current project where you can choose to launch another view.

Select Configuration Opens the **Select a View Configuration** dialog box where you can specify which files to check out in the current view based on a label, promotion state, or a date and time.

Properties Opens the **View Properties** dialog box for the current view.

Access Rights Opens the **View Access Rights** dialog box for the current view.

Labels Opens the **Labels** dialog box where you can create, delete, or freeze (lock) a view label or a revision label or review its properties.

Promotion Opens the **Promotion** dialog box where you can create, edit, delete, reorder, or set access rights for promotion states. A promotion state is a way of grouping files as they progress through different stages to completion (for example, Develop, Test, and Production).

Compare/Merge Opens the **View Compare/Merge Wizard** where you can compare or merge different views.

Compare/Merge Active Process Item Opens the **View Compare/Merge Wizard** where you can scope a view compare merge session with a process item in a different view than your current process.

Folder Tree Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New Starts the **New Folder Wizard**, and creates a new folder in the current view.

Add to View Adds the selected local folder to the current view.

Create Working Folders Creates the path to a working folder on your hard drive. Otherwise, if the path to a working folder does not exist on your hard drive, you must create it manually before adding files or checking in files from that working folder.

Delete Local Folders Deletes the specified local folder and any subfolders. This action does not delete any folder or subfolder from the StarTeam repository.

Open Local Folder Opens a local folder in a file browser.

Show Not-In-View Folders Works as a toggle. Choose to make not-in-view folders visible to the client. Not-in-view folders reside in your local working folder, but they have not been added to the StarTeam repository.

Lock/Unlock Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.

Properties Opens the **<<Component Type>> Properties** dialog box for the selected item.

Labels Opens the **Labels** dialog box where you can assign labels to and/or removes labels from folders.

Advanced > Behavior Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.

Advanced > References Opens the **Folder References** dialog box where you can view the relationships between an original folder or item and the others based on it.

Advanced > Access Rights Opens the **Folder Access Rights** dialog box for the selected folder.

Advanced > View Compare/Merge Opens the **View Compare/Merge Wizard**.

Links > Create Link Start a link from a selected item to another item.

Links > Complete Link Complete the link from an item to the selected item.

Links > Cancel Link Stop the process of creating a link.

Copy URL to Clipboard Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.

Folder Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New Starts the **New Folder Wizard**, and creates a new folder in the current view.

Add to View Adds the selected local folder to the current view.

Create Working Folders Creates the path to a working folder on your hard drive. Otherwise, if the path to a working folder does not exist on your hard drive, you must create it manually before adding files or checking in files from that working folder.

Delete Local Folders Deletes the specified local folder and any subfolders. This action does not delete any folder or subfolder from the StarTeam repository.

Open Local Folder Opens a local folder in a file browser.

Properties Opens the **<<Component Type>> Properties** dialog box for the selected item.

Copy URL to Clipboard Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.

Send To E-mail a copy of the item properties to one or more team members.

Save Shortcut Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.

Lock/Unlock	Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.
Compare Properties	Compares the properties of two selected items.
Labels > New	Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).
Labels > Attach	Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).
Labels > Detach	Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).
Advanced > Export	Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.
Advanced > Behavior	Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.
Advanced > Access Rights	Opens the **<<Item Type>> Access Rights** dialog box for the selected item.
Advanced > Component Access Rights	Opens the **<<Item Type>> Component Access Rights** dialog box.
Advanced > View Compare/Merge	Opens the **View Compare/Merge Wizard**.
Select > All	Selects all items displayed in the upper pane.
Select > By Query	Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.
Select > By Label	Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.
All Descendants	Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.
Delete	Deletes the item from the StarTeam repository.
Find	Finds all items containing text that matches the specified text string.
Find Next	Finds the next item containing text that matches the specified text string.
Find Previous	Finds the previous item containing text that matches the specified text string.
Filters > Show Fields	Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.
Filters > Sort and Group	Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.
Filters > Queries	Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.
Filters > Save Current Settings	Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.
Filters > Reset Current Settings	Returns to the current default filter as defined on the server.

Filters > Filters	Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.
Reports	Opens the **Reports** dialog box where you can create a report using the selected item.
Charts > Simple	Creates a simple chart. A simple chart contains only one series.
Charts > Correlation	Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.
Charts > Time-series	Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.
Links > Create Link	Start a link from a selected item to another item.
Links > Complete Link	Complete the link from an item to the selected item.
Links > Cancel Link	Stop the process of creating a link.
Linked Files > Check In All	Checks in any files from the current view linked to the item.
Linked Files > Check Out All	Checks out any files from the current view linked to the item.
Linked Files > Select All	Selects all the files associated with a linked item in the current view.
Linked Files > Add to Selection	Opens the upper pane to the **File** pane, activates the **All Descendants** view, and adds all the files linked to the item to the existing file selection.

File Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

Open	Opens the selected file in its associated application or starts an executable file. If you do not have a copy of the file locally, the application prompts you to save a copy in your local working directory.
Edit	Edits the selected file in the default text editor used by your system or by the editor specified in the **Alternate Applications** dialog box.
Open Containing Folder	Opens the local folder that contains the file in a file browser.
Properties	Opens the **<<Component Type>> Properties** dialog box for the selected item.
Copy URL to Clipboard	Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.
Send To	E-mail a copy of the item properties to one or more team members.
Save Shortcut	Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.
Add Files	Adds new files to the current project view.

Check In	Checks the selected files into the current project view.
Check Out	Copies the selected files to the appropriate working folders.
Check Out All	Checks out all the files in the branch of the application folder hierarchy that has the selected folder as its root folder.
Update Status	Updates the status of the selected files. If no files are selected, all file statuses are updated.
Lock/Unlock	Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.
Compare Properties	Compares the properties of two selected items.
Compare Properties	Compares the properties of two selected files.
Compare Contents	Opens File Compare/Merge.
Labels > New	Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).
Labels > Attach	Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).
Labels > Detach	Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).
Advanced > Export	Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.
Advanced > Behavior	Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.
Advanced > Access Rights	Opens the **<<Item Type>> Access Rights** dialog box for the selected item.
Advanced > Component Access Rights	Opens the **<<Item Type>> Component Access Rights** dialog box.
Advanced > Customize	Opens the **Customize** dialog box where you can create custom fields that become item properties.
Advanced > View Compare/Merge	Opens the **View Compare/Merge Wizard**.
Select > All	Selects all items displayed in the upper pane.
Select > By Query	Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.
Select > By Label	Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.
Expand All	Expands all the groups in the list.
Collapse All	Collapses all the groups in the list.
All Descendants	Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.
Flag	Flags an item. Only you can see and remove your flags. When an item is flagged, `Yes` displays in the **Flag** column in the upper pane.

Note: This column does not display by default. Choose **<<Component Type>>** > **Filters** > **Show Fields** to add the column to the upper pane.

Remove Flag Removes the flag from an item.

Delete Deletes the item from the StarTeam repository.

Find Finds all items containing text that matches the specified text string.

Find Next Finds the next item containing text that matches the specified text string.

Find Previous Finds the previous item containing text that matches the specified text string.

Filters > Show Fields Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.

Filters > Sort and Group Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.

Filters > Queries Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.

Filters > Reset Current Settings Returns to the current default filter as defined on the server.

Filters > Filters Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.

Reports Opens the **Reports** dialog box where you can create a report using the selected item.

Charts > Simple Creates a simple chart. A simple chart contains only one series.

Charts > Distribution Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.

Charts > Correlation Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.

Charts > Time-series Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.

Links > Create Link Start a link from a selected item to another item.

Links > Complete Link Complete the link from an item to the selected item.

Links > Cancel Link Stop the process of creating a link.

Linked Files > Check In All Checks in any files from the current view linked to the item.

Linked Files > Check Out All Checks out any files from the current view linked to the item.

Linked Files > Select All Selects all the files associated with a linked item in the current view.

Linked Files > Add to Selection	Opens the upper pane to the **File** pane, activates the **All Descendants** view, and adds all the files linked to the item to the existing file selection.

Change Request Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New	Creates a new item.
Properties	Opens the **<<Component Type>> Properties** dialog box for the selected item.
Copy URL to Clipboard	Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.
Send To	E-mail a copy of the item properties to one or more team members.
Save Shortcut	Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.
Lock/Unlock	Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.
Compare Properties	Compares the properties of two selected items.
Labels > New	Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).
Labels > Attach	Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).
Labels > Detach	Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).
Advanced > Export	Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.
Advanced > Behavior	Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.
Advanced > Access Rights	Opens the **<<Item Type>> Access Rights** dialog box for the selected item.
Advanced > Component Access Rights	Opens the **<<Item Type>> Component Access Rights** dialog box.
Advanced > Customize	Opens the **Customize** dialog box where you can create custom fields that become item properties.
Advanced > View Compare/Merge	Opens the **View Compare/Merge Wizard**.
Select > All	Selects all items displayed in the upper pane.
Select > By Query	Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.
Select > By Label	Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.

Expand All
Expands all the groups in the list.

Collapse All
Collapses all the groups in the list.

All Descendants
Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.

Mark as Read
Displays the selected items in regular font (not bold), which indicates that they have been read.

Mark as Unread
Displays the selected items in bold font, which indicates that they have not been read.

Mark Thread as Read
Displays the entire thread in regular type, which indicates that the items it contains have been read.

Mark Thread as Unread
Displays the entire thread in boldface type, which indicates that the items it contains have not been read.

Flag
Flags an item. Only you can see and remove your flags. When an item is flagged, `Yes` displays in the **Flag** column in the upper pane.

Note: This column does not display by default. Choose **<<Component Type>>** > **Filters** > **Show Fields** to add the column to the upper pane.

Remove Flag
Removes the flag from an item.

Delete
Deletes the item from the StarTeam repository.

Find
Finds all items containing text that matches the specified text string.

Find Next
Finds the next item containing text that matches the specified text string.

Find Previous
Finds the previous item containing text that matches the specified text string.

Filters > Show Fields
Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.

Filters > Sort and Group
Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.

Filters > Queries
Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings
Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.

Filters > Reset Current Settings
Returns to the current default filter as defined on the server.

Filters > Filters
Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.

Reports
Opens the **Reports** dialog box where you can create a report using the selected item.

Charts > Simple
Creates a simple chart. A simple chart contains only one series.

Charts > Distribution
Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.

Charts > Correlation
Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.

Charts > Time-series
Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.

Links > Create Link
Start a link from a selected item to another item.

Links > Complete Link
Complete the link from an item to the selected item.

Links > Cancel Link
Stop the process of creating a link.

Set Active Process Item > Current View
Designates the selected item in the current view as the active process item. Active process items are used when you add items to a view or check them in.

Set Active Process Item > Select View
Designates the selected item in an alternate view as the active process item. Active process items are used when you add items to a view or check them in.

Clear Active Process Item
Changes the selected item from `Active` to `Inactive`. `Active` process items are used when you add items to a view or check them in.

Requirement Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New
Creates a new item.

New Child Requirement
Creates a requirement that is the child of another requirement.

Properties
Opens the **<<Component Type>> Properties** dialog box for the selected item.

Copy URL to Clipboard
Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.

Send To
E-mail a copy of the item properties to one or more team members.

Save Shortcut
Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.

Lock/Unlock
Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.

Compare Properties
Compares the properties of two selected items.

Labels > New
Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).

Labels > Attach
Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).

Labels > Detach
Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).

Advanced > Export
Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.

Advanced > Behavior	Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.
Advanced > Access Rights	Opens the **<<Item Type>> Access Rights** dialog box for the selected item.
Advanced > Component Access Rights	Opens the **<<Item Type>> Component Access Rights** dialog box.
Advanced > Customize	Opens the **Customize** dialog box where you can create custom fields that become item properties.
Advanced > View Compare/Merge	Opens the **View Compare/Merge Wizard**.
Select > All	Selects all items displayed in the upper pane.
Select > By Query	Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.
Select > By Label	Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.
Expand All	Expands all the groups in the list.
Collapse All	Collapses all the groups in the list.
All Descendants	Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.
Mark as Read	Displays the selected items in regular font (not bold), which indicates that they have been read.
Mark as Unread	Displays the selected items in bold font, which indicates that they have not been read.
Mark Thread as Read	Displays the entire thread in regular type, which indicates that the items it contains have been read.
Mark Thread as Unread	Displays the entire thread in boldface type, which indicates that the items it contains have not been read.
Flag	Flags an item. Only you can see and remove your flags. When an item is flagged, Yes displays in the **Flag** column in the upper pane. **Note:** This column does not display by default. Choose **<<Component Type>>** > **Filters** > **Show Fields** to add the column to the upper pane.
Remove Flag	Removes the flag from an item.
Delete	Deletes the item from the StarTeam repository.
Find	Finds all items containing text that matches the specified text string.
Find Next	Finds the next item containing text that matches the specified text string.
Find Previous	Finds the previous item containing text that matches the specified text string.
Filters > Show Fields	Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.
Filters > Sort and Group	Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.
Filters > Queries	Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings	Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.
Filters > Reset Current Settings	Returns to the current default filter as defined on the server.
Filters > Filters	Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.
Reports	Opens the **Reports** dialog box where you can create a report using the selected item.
Charts > Simple	Creates a simple chart. A simple chart contains only one series.
Charts > Distribution	Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.
Charts > Correlation	Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.
Charts > Time-series	Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.
Links > Create Link	Start a link from a selected item to another item.
Links > Complete Link	Complete the link from an item to the selected item.
Links > Cancel Link	Stop the process of creating a link.
Set Active Process Item > Current View	Designates the selected item in the current view as the active process item. Active process items are used when you add items to a view or check them in.
Set Active Process Item > Select View	Designates the selected item in an alternate view as the active process item. Active process items are used when you add items to a view or check them in.
Clear Active Process Item	Changes the selected item from `Active` to `Inactive`. `Active` process items are used when you add items to a view or check them in.

Task Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New	Creates a new item.
New Subtask	Creates a task that is the child of another task.
Add Work	Adds a work record to the selected task. You can add work only to tasks that have no sub-tasks.
Properties	Opens the **<<Component Type>> Properties** dialog box for the selected item.

Copy URL to Clipboard Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.

Send To E-mail a copy of the item properties to one or more team members.

Save Shortcut Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.

Save Shortcut Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.

Lock/Unlock Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.

Compare Properties Compares the properties of two selected items.

Labels > New Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).

Labels > Attach Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).

Labels > Detach Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).

Advanced > Export Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.

Advanced > Behavior Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.

Advanced > Access Rights Opens the **<<Item Type>> Access Rights** dialog box for the selected item.

Advanced > Component Access Rights Opens the **<<Item Type>> Component Access Rights** dialog box.

Advanced > Customize Opens the **Customize** dialog box where you can create custom fields that become item properties.

Advanced > View Compare/Merge Opens the **View Compare/Merge Wizard**.

Select > All Selects all items displayed in the upper pane.

Select > By Query Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.

Select > By Label Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.

Expand All Expands all the groups in the list.

Collapse All Collapses all the groups in the list.

All Descendants Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.

Mark as Read Displays the selected items in regular font (not bold), which indicates that they have been read.

Mark as Unread Displays the selected items in bold font, which indicates that they have not been read.

Mark Thread as Read Displays the entire thread in regular type, which indicates that the items it contains have been read.

Mark Thread as Unread Displays the entire thread in boldface type, which indicates that the items it contains have not been read.

Flag Flags an item. Only you can see and remove your flags. When an item is flagged, `Yes` displays in the **Flag** column in the upper pane.

Note: This column does not display by default. Choose **<<Component Type>>** > **Filters** > **Show Fields** to add the column to the upper pane.

Remove Flag Removes the flag from an item.

Delete Deletes the item from the StarTeam repository.

Find Finds all items containing text that matches the specified text string.

Find Next Finds the next item containing text that matches the specified text string.

Find Previous Finds the previous item containing text that matches the specified text string.

Filters > Show Fields Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.

Filters > Sort and Group Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.

Filters > Queries Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.

Filters > Reset Current Settings Returns to the current default filter as defined on the server.

Filters > Filters Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.

Reports Opens the **Reports** dialog box where you can create a report using the selected item.

Charts > Simple Creates a simple chart. A simple chart contains only one series.

Charts > Distribution Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.

Charts > Correlation Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.

Charts > Time-series Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.

Links > Create Link Start a link from a selected item to another item.

Links > Complete Link Complete the link from an item to the selected item.

Links > Cancel Link	Stop the process of creating a link.
Set Active Process Item > Current View	Designates the selected item in the current view as the active process item. Active process items are used when you add items to a view or check them in.
Set Active Process Item > Select View	Designates the selected item in an alternate view as the active process item. Active process items are used when you add items to a view or check them in.
Clear Active Process Item	Changes the selected item from `Active` to `Inactive`. `Active` process items are used when you add items to a view or check them in.

Topic Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

New	Creates a new item.
Respond	Send a reply to a previous message.
Properties	Opens the **<<Component Type>> Properties** dialog box for the selected item.
Copy URL to Clipboard	Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.
Send To	E-mail a copy of the item properties to one or more team members.
Save Shortcut	Creates a shortcut to the selected item. You can use the shortcut to start the application, open the project view in the configuration in which the shortcut was created, and open the **Properties** dialog box for the selected item.
Lock/Unlock	Opens the **Set My Lock Status** dialog box where you can specify the lock status (`Unlocked`, `Exclusive`, or `Non-Exclusive`) for the folder or break an existing lock on an item.
Compare Properties	Compares the properties of two selected items.
Labels > New	Opens the **Attach a New Revision Label** dialog box where you can create a new revision label and attach it to the selected item or item(s).
Labels > Attach	Opens the **Attach a Label** dialog box where you can add an existing label to the selected items(s).
Labels > Detach	Opens the **Detach a Label** dialog box where you can remove a label from the selected items(s).
Advanced > Export	Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.
Advanced > Behavior	Opens the **<<Item Type>> Behavior** dialog box where you can view or change the behavior of the selected item.
Advanced > Access Rights	Opens the **<<Item Type>> Access Rights** dialog box for the selected item.
Advanced > Component Access Rights	Opens the **<<Item Type>> Component Access Rights** dialog box.

Advanced > Customize Opens the **Customize** dialog box where you can create custom fields that become item properties.

Advanced > View Compare/Merge Opens the **View Compare/Merge Wizard**.

Select > All Selects all items displayed in the upper pane.

Select > By Query Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.

Select > By Label Opens the **Select a Label** dialog box where you can specify a label that selects all items displayed in the upper pane that have the specified label.

Expand All Expands all the groups in the list.

Collapse All Collapses all the groups in the list.

All Descendants Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.

Mark as Read Displays the selected items in regular font (not bold), which indicates that they have been read.

Mark as Unread Displays the selected items in bold font, which indicates that they have not been read.

Mark Thread as Read Displays the entire thread in regular type, which indicates that the items it contains have been read.

Mark Thread as Unread Displays the entire thread in boldface type, which indicates that the items it contains have not been read.

Flag Flags an item. Only you can see and remove your flags. When an item is flagged, `Yes` displays in the **Flag** column in the upper pane.

> **Note:** This column does not display by default. Choose **<<Component Type>>** > **Filters** > **Show Fields** to add the column to the upper pane.

Remove Flag Removes the flag from an item.

Delete Deletes the item from the StarTeam repository.

Find Finds all items containing text that matches the specified text string.

Find Next Finds the next item containing text that matches the specified text string.

Find Previous Finds the previous item containing text that matches the specified text string.

Filters > Show Fields Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.

Filters > Sort and Group Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.

Filters > Queries Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.

Filters > Reset Current Settings Returns to the current default filter as defined on the server.

Filters > Filters	Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.
Reports	Opens the **Reports** dialog box where you can create a report using the selected item.
Charts > Simple	Creates a simple chart. A simple chart contains only one series.
Charts > Distribution	Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.
Charts > Correlation	Creates a correlation chart. A correlation chart shows the relationship or degree of relationship between numeric values in several series.
Charts > Time-series	Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/date field.
Links > Create Link	Start a link from a selected item to another item.
Links > Complete Link	Complete the link from an item to the selected item.
Links > Cancel Link	Stop the process of creating a link.
Set Active Process Item > Current View	Designates the selected item in the current view as the active process item. Active process items are used when you add items to a view or check them in.
Set Active Process Item > Select View	Designates the selected item in an alternate view as the active process item. Active process items are used when you add items to a view or check them in.
Clear Active Process Item	Changes the selected item from `Active` to `Inactive`. `Active` process items are used when you add items to a view or check them in.

Audit Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

Copy URL to Clipboard	Places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.
Send To	E-mail a copy of the item properties to one or more team members.
Export	Opens the **Export** dialog box where you can export the data displayed in the upper pane for use in a spreadsheet, database, or another application.
Select > All	Selects all items displayed in the upper pane.
Select > By Query	Opens the **Select Query** dialog box where you can specify a query that selects all items displayed in the upper pane that match the specified query.
Expand All	Expands all the groups in the list.
Collapse All	Collapses all the groups in the list.

All Descendants Displays all items associated with the folder selected from the folder hierarchy and all of its child folders. When this option is not selected, the upper pane displays only those items associated with the selected folder.

Find Finds all items containing text that matches the specified text string.

Find Next Finds the next item containing text that matches the specified text string.

Find Previous Finds the previous item containing text that matches the specified text string.

Filters > Show Fields Opens the **Show Fields** dialog box where you can select which column headers to display in the upper pane.

Filters > Sort and Group Opens the **Sort and Group** dialog box where you can specify how to sort and group the data in the upper pane based on up to four item properties.

Filters > Queries Opens the **Queries** dialog box where you can create or apply a query. Items that match the query become the rows displayed in the upper pane.

Filters > Save Current Settings Opens the **Save Current Settings** dialog box where you can save the current column headers, query, and method of sorting and grouping as a filter, which can be reapplied in the future.

Filters > Reset Current Settings Returns to the current default filter as defined on the server.

Filters > Filters Opens the **Filters** dialog box where you can display the filters available for this component. This option allows you to select a filter or create a new one. An asterisk preceding a filter name in the filter list indicates that you have changed the contents of the upper pane since the filter was applied. For example, you may have displayed additional fields, grouped the items differently, or applied another query.

Reports Opens the **Reports** dialog box where you can create a report using the selected item.

Charts > Distribution Creates a distribution chart. A distribution chart is the same as a pie chart. Each wedge indicates what fraction of the whole a group represents.

Charts > Time-series Creates a time-series chart. A time-series chart is a line chart that shows the number of items that have the same day, week, or month in the specified time/ date field.

Tools Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

Personal Options Opens the **Personal Options** dialog box where you can customize the application.

My Account Opens the **My Account** dialog box where you can edit your account information, change your password, or review the logged-on user's group membership information.

Show Advanced Menus Allows you to control whether standard or advanced menu items display in the component menu selected from the Cross-Platform Client Main Menu. **Show Advanced Menus** is turned on by default and a check box displays when the option is turned on.

StarTeam Log Displays the latest information from the client log, primarily for diagnostic purposes. The log can display error messages that occur during use of the application,

	summaries of each operation performed, details about each operation performed, and StarTeamMPX events.
Convert to Native Format	Converts files saved in the Native-I vault structure to the Native-II vault structure.
File Annotation	Shows historical information about changes made to text files.
Compare	Opens the embedded version of File Compare/Merge.

Window Menu

This menu is always available on the main window, regardless of the component selected from the upper pane. Depending upon the privileges assigned to you and the activities you need to perform, some options may not be enabled.

Cascade	Arranges open windows so that they overlap but all their titles are visible.
Tile Horizontally	Arranges open windows horizontally without overlapping them.
Tile Vertically	Arranges the open windows vertically without overlapping them.
Arrange Icons	Orders the icons for minimized windows.
Refresh	Refreshes the contents of the project window.
Component Refresh	Refreshes the contents of the upper pane without collapsing open groups of items. For example, if the **File** tab has been selected, this option refreshes the **File** list.
Component Refresh/ Collapse	Refreshes the contents of the upper pane and simultaneously collapse open groups of items.
Foreign Refresh	Refreshes third-party projects or archives, such as Microsoft Visual SourceSafe projects and PVCS archive files.
Recent Views	Displays a list of the most recently accessed project view windows. A checkmark displays in front of the currently displayed project view. To display a different project view, click its name.

Help Menu

The **Help** menu is always available on the main window, regardless of the component selected from the upper pane.

Help Topics	Opens the help system for StarTeam. Allows you to view help topics on the application's features.
About	Displays release and copyright information, and provides information about the Java virtual machine and the location of certain classes on your workstation.

Client Keyboard Shortcuts

Below is a list of keyboard shortcuts to use in the Cross-Platform Client.

Cancel Link Creation	**Escape**
Check In	**Ctrl+I**
Check In and Unlock	**Ctrl+U**
Check Out	**Ctrl+G**

Check Out and Unlock	**Ctrl+O**
Component Refresh	**F5**
Component Refresh/Collapse	**Ctrl+F5**
Delete	**Delete**
Find	**Ctrl+F**
Find Next	**F3**
Find Previous	**Shift+F3**
Flag	**Ctrl+F2**
Remove Flag	**Ctrl+Shift+F2**
Foreign Refresh	**F6**
Launch File Annotation tool	**Alt+F1**
Launch File Compare tool	**Alt+F2**
Launch Help	**F1**
Lock/Unlock	**Ctrl+L**
Project New	**Ctrl+N**
Project Open	**Ctrl+P**
Redo character in Text Edit	**Ctrl+Y**
Select All	**Ctrl+A**
Show Item Properties	**Alt+Enter**
Undo character in Text Edit	**Ctrl+Z**
Window Refresh	**Shift+F5**

Note: On the Linux platform, **Alt+F1**, **Alt+F2**, **Alt+F5**, and **Alt+F9** are designated as KDE global shortcuts. In order to use these shortcuts to launch the File Annotation and File Compare tools in the client, you must first disable them in the KDE application.

Change Perspective UI

The **Change Perspective** gives you a view of all the View Compare/Merge change packages which have been created, and either are committed or are still uncommitted. The **Change Perspective** lets you manage change perspectives, review all the information about change packages, review what changes have been made to in each change package, or what changes are being proposed for an uncommitted change package. Managers can review change packages before they are committed, and if appropriate, make additional changes and commit the sessions themselves.

All the changes in a committed change package can be *Replayed* (applied to another view) using the menu drag and drop, saving the effort and time of re-creating from scratch the same changes in a new VCM change session for another view.

Note: The **Change Perspective** is actually an alternate view window in the main Cross-Platform Client which can be displayed by clicking on the **Show Change Perspective** icon on the right side of the toolbar. Clicking the **Show Content Perspective** icon switches you back to the familiar Cross-Platform Client. Each perspective has its own menu and set of toolbar buttons.

Change Perspective Layouts

When you switch to the **Change Perspective**, you see the default “side by side” layout with the changes packages listed on the left, and the **Details** pane on the right. If you have performed and saved any View Compare/Merge sessions prior to showing the **Change Perspective**, you will see them in the change perspectives list. Which change packages are displayed depends on the filter you select at the top of the list.

StarTeam gives you the option of using one of the two following layouts for the Change Perspective: the “side by side” vertical arrangement, or the “top over bottom” horizontal arrangement.

Side by Side Layout

The “side by side” layout lists the change packages in a column on the left, and the **Detail**, **History**, **Label**, **Link**, **Change**, and **Replay** views in a pane on the right.

The **Change Perspective** uses some of the regular views from the **Content Perspective**, but it also has two different views for Change Perspective and Change actions in particular: The **Change**, and **Replay** views. These are discussed in more detail later.

In the “side by side” layout, clicking a toolbar button at the top right of the view pane switches the view. Those view toolbar buttons correspond to the tabs in the “top over bottom” view, and have tool tips to show you their functions.

Top over Bottom Layout

The “top over bottom” layout shows the change packages in the top half of the window, and the **Detail**, **History**, **Label**, **Link**, **Change**, and **Replay** views in tabbed panes at the bottom, much like the typical Cross-Platform Client **Content Perspective**.

Note: To switch to the “top over bottom” layout, click the **Layout Top Over Bottom** button on the main toolbar.

Change Perspective Toolbars, Tabs, and Menus

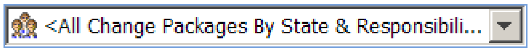
Each **Change Perspective** layout uses either toolbars, tabs, and menus to perform the different change package tasks, or to see different views. As with StarTeam in general, each toolbar button has a corresponding menu item, either on the **Change Package** menu, or on the context menu inside the **Change Perspective**.

Toolbars and Tabs

The “side by side” **Change Perspective** layout uses toolbars at the top of each side to perform functions or change views.

Each side in this layout has its own toolbar. On the left, above the list of change sessions, there are two icons and the filter list.

Properties	Displays the **Change Properties** dialog box for the selected change package. You can view the change package properties, or if you have the proper access rights, you can modify the properties.
Open (in VCM)	Opens the selected change package in a VCM session. If the change package is new and uncommitted, you can resolve conflicts and commit the session. However, if the selected change package has already been committed, it opens in the VCM session in Read-Only mode. No changes can be made once the session is committed.


<All Change Packages By State & Responsibili...

Filter

Displays the list of change package sessions according to the selected filters. As with the rest of StarTeam, you can create new filters and queries for the **Change Perspective**.

Note: The toolbar items on the left move up to the main Cross-Platform Client window in the “top over bottom” layout of the **Change Perspective**

Displays all the details for the selected change package in the VCM session in which it was created. A `changepackage.details.html` file is installed by default and utilized by the **Detail Viewer** which is fully customizable. See “*Customizing the Detail Pane*."

History View

Detail View - Displays the historical information for the selected change package.

Labels View - Displays the branch and revision labels for the selected change package.

Displays the link information for the selected change package.

Changes View

Links View - Displays all the changes made in the selected committed change package. You can double-click on either a new or committed change package to see those changes in VCM session. You can continue working on the change package in VCM if it has not been committed, however, a committed change package will be read-only and cannot be changed.

Replay View - Displays all the possible views to which a change package can be replayed (the same changes reapplied to another view). simply by dragging the selected change package to the desired view in the Replay list. If the Replay is allowed, the icon displays a plus with it. If the Replay to the target view is not allowed, the cursor changes to a circle with a diagonal line.

Notice that when you switch to the “top over bottom” layout, the following occurs:

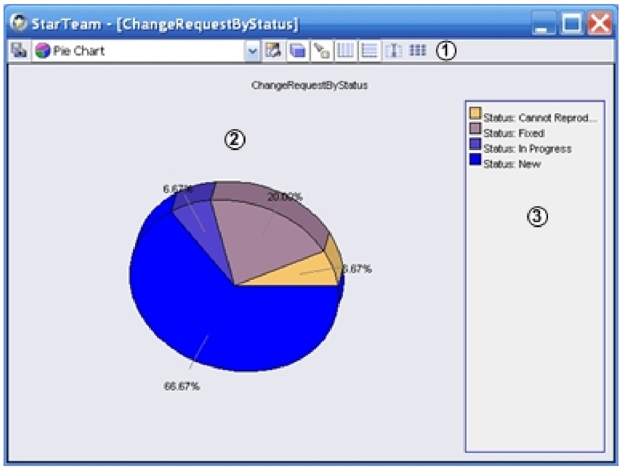
- The toolbar icons and **Filter** list move up to the main toolbar.
- The toolbar buttons for the different views that were on the right pane of the “side by side” view correspond exactly to the tabbed panes in the lower half of the “top over bottom” layout.

They perform exactly the same functions in either layout.

Chart Window

Chart Window UI

The Cross-Platform Client enables you to generate charts from items in the upper pane.



① Chart Window Toolbar ② Chart Display Area ③ Chart Legend

Toolbar After generating a chart the **Chart** window opens displaying the chart of the items specified. The **Chart window** toolbar enables you to :

- Save charts as .jpg files.
- Select a different chart type, such as an area chart or bar chart.
- Edit chart colors.
- View charts in 2–D or 3–D.
- Optionally show chart legends or horizontal or vertical grid lines.
- Edit chart titles.
- Optionally show chart data in a table format.

Display Area The chart display area presents your chart data. You can select a different chart type using the drop-down list provided in the chart window toolbar.

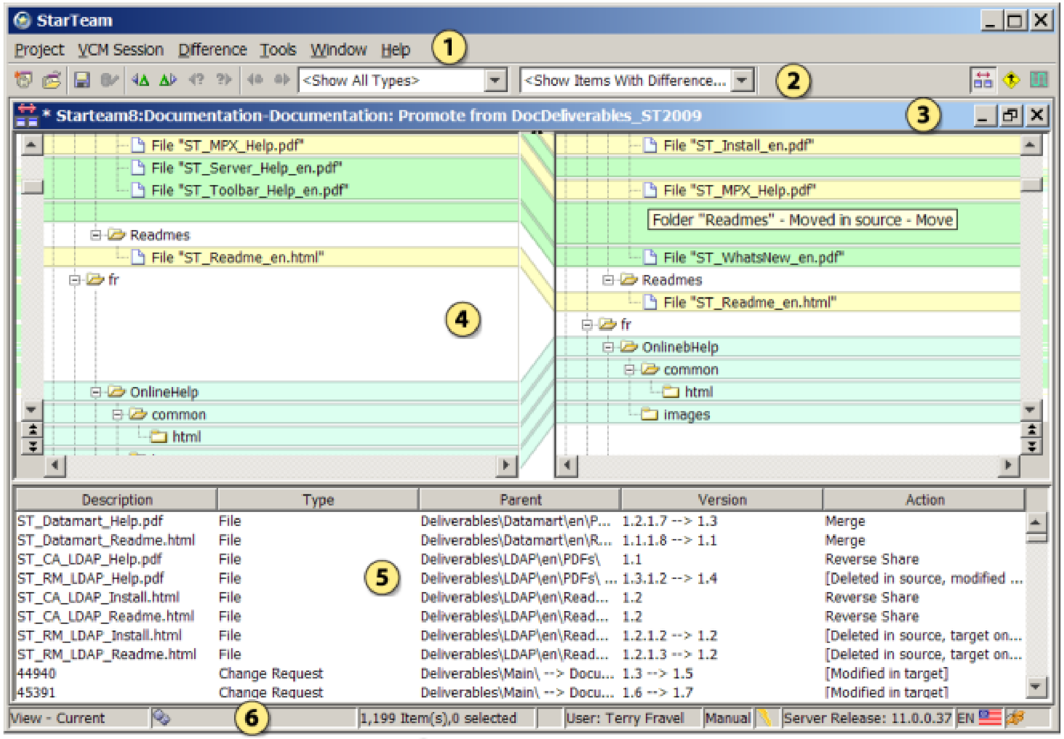
Legend The chart legend displays by default. You can optionally display the chart legend (toggle on or off) using the **Toggle legend** toolbar button in the chart window toolbar.

View Compare/Merge

View Compare/Merge (VCM) is the tool you use to compare two views or subsets of two views, referred to as the source and the target. As part of the process, you can merge item differences in the source view to the target view.

The View Compare/Merge UI is available in the StarTeam Cross-Platform Client and opens after running the **View Compare/Merge Wizard**.

View Compare/Merge UI


StarTeam
Project VCM Session Difference Tools Window Help
1
<Show All Types>
<Show Items With Difference...
2
* Starteam8:Documentation-Documentation: Promote from DocDeliverables_ST2009
3
File "ST_MPX_Help.pdf"
File "ST_Server_Help_en.pdf"
File "ST_Toolbar_Help_en.pdf"
Readmes
File "ST_Readme_en.html"
fr
OnlineHelp
common
html
File "ST_Install_en.pdf"
File "ST_MPX_Help.pdf"
Folder "Readmes" - Moved in source - Move
File "ST_WhatsNew_en.pdf"
Readmes
File "ST_Readme_en.html"
fr
OnlinebHelp
common
html
images
4
5
View - Current
6
1,199 Item(s),0 selected
User: Terry Fravel
Manual
Server Release: 11.0.0.37
EN

Description	Type	Parent	Version	Action
ST_Datamart_Help.pdf	File	Deliverables\Datamart\en\P...	1.2.1.7 --> 1.3	Merge
ST_Datamart_Readme.html	File	Deliverables\Datamart\en\R...	1.1.1.8 --> 1.1	Merge
ST_CA_LDAP_Help.pdf	File	Deliverables\LDAP\en\PDFs\	1.1	Reverse Share
ST_RM_LDAP_Help.pdf	File	Deliverables\LDAP\en\PDFs\ ...	1.3.1.2 --> 1.4	[Deleted in source, modified ...
ST_CA_LDAP_Install.html	File	Deliverables\LDAP\en\Read...	1.2	Reverse Share
ST_CA_LDAP_Readme.html	File	Deliverables\LDAP\en\Read...	1.2	Reverse Share
ST_RM_LDAP_Install.html	File	Deliverables\LDAP\en\Read...	1.2.1.2 --> 1.2	[Deleted in source, target on...
ST_RM_LDAP_Readme.html	File	Deliverables\LDAP\en\Read...	1.2.1.3 --> 1.2	[Deleted in source, target on...
44940	Change Request	Deliverables\Main\ --> Docu...	1.3 --> 1.5	[Modified in target]
45391	Change Request	Deliverables\Main\ --> Docu...	1.6 --> 1.7	[Modified in target]

1 VCM Menu	3 VCM Title Bar	5 Lower Detail Pane
2 VCM Toolbar	4 Upper Comparison Pane	6 Status Bar

The View Compare/Merge window provides three perspectives for comparing and merging views, and for resolving item differences.

- The **Compare Perspective**
- The **Merge Perspective**
- The **Test Perspective**

The image above displays the **Compare Perspective**.

VCM Menu

The View Compare/Merge menu consists of a subset of the main client menus, plus two additional menus specific to View Compare/Merge:

- **VCM Session**, which enables you to view a View Compare/Merge session summary, change perspectives, save, export, or delete your session, display a session report, and commit your changes.
- **Difference**, which enables you to view item properties, navigate to items with differences, merge file contents, and specify the merge action for an item.

VCM Toolbar

Frequently used View Compare/Merge menu commands corresponding to the **VCM Session** and **Difference** menus have corresponding buttons on the toolbar. Tool tips display when you hover your mouse over the toolbar buttons. Filters specific to View Compare/Merge and the current perspective display in the **Filters** drop-down list on the **VCM Toolbar**.

Note: The **Refresh** button on the toolbar in the Merge Perspective does not change the snapshot of the source/target in use. It only refreshes the display with the View Compare/Merge session data that was already gathered. For instance, it might re-group items based on changes made by the user in the View Compare/Merge window. For example, if you go back to the target view and move a file, when you return to the merge window and selected **Window** > **Refresh** , the moved file will still be shown in its original location.

Upper Pane

The upper pane serves a different function in each perspective:

- In the **Compare Perspective**, the upper pane displays two trees comparing the items in the source and target views. The tree nodes are highlighted to show the differences for all item types in the views or folders being compared.
- In the **Merge Perspective**, the upper pane displays the individual items on component tabs. The component tabs display View Compare/Merge icons indicating there are resolved or unresolved conflicts of that item type in the selected folder. Icons also display in the **Merge Status** property column for each item with resolved or unresolved differences. See the topic *VCM Icons* for an image and description of each icon.
- In the **Test Perspective**, the upper pane shows the target check-out displayed in the same way as the **Merge Perspective**.

Lower Pane

The lower pane serves a different function for each perspective:

Compare Perspective

The lower pane shows the details and merge action for the items in the View Compare/Merge session. You can filter which items display in this perspective, and you can change the merge action for items in this pane.

Merge Perspective

The lower pane contains three tabs. An item **Property** tab displays the properties from the source and target views for comparison, and enables you to edit them. Two **File Comparison** tabs compare the contents of the source and target files, or the target and merged files.

Test Perspective

Contains the same information tabs as the main client window.

View Compare/Merge Status Icons

The status of folders and items in a View Compare/Merge session is represented in the **Merge Perspective** by icons which are explained in the tables below. Icons appear on or beside folders, items, and on component tabs. Icons change dynamically when conflicts are resolved.

When you hover the mouse cursor over an icon in the upper pane, a menu appears from which you can choose a merge action. Tool tips display for folder icons.

📁 This folder contains compared items that remain unchanged.

📁? This folder contains compared items, some of which have unresolved changes called merge conflicts. Merge conflicts are differences in file contents or property values that require manual intervention.

📁✓ This folder contains compared items, none of which have unresolved changes. No manual intervention is necessary.

📁X This folder contains no compared items or folders.

The folder icons are displayed in the left pane beside each folder in the hierarchy tree of a View Compare/Merge session in the **Merge Perspective**. The folder icons in a View Compare/Merge session represent what is in the currently selected component tab in the upper pane. For example, in the situation where you have resolved all the file conflicts in a particular folder, but you still have unresolved change requests in that folder, the following will happen:

- When you click the **Files** tab, the folder icon will have a check mark, indicating that all file conflicts in that folder have been resolved.

- When you click the **Change Request** tab, the folder icon will have a question mark, indicating that the folder has unresolved change requests.

No icon	Item is included in the comparison and is unchanged.
✓	Item resolved by View Compare/Merge. You can override the default action with a different action.
?	Item has differences and is unresolved. You must resolve the differences manually.
✗	Item is not included in the comparison.
△	Item is changed. This is used in Compare-only.

Item status icons are displayed in the upper pane of the **Merge Perspective** for each item with differences or conflicts. They are also displayed on each component tab that contains items with differences. Item icons appear in the **Merge Status** and **Merge Action** columns, and on individual property columns when properties are in conflict.

Logging on to and off of a Server

This section contains tasks related to logging on to and off of a StarTeam server and how to open a project.

Logging on to StarTeam Server and Creating or Opening a Project

Before you can create a new project or open an existing project, you must select a server configuration for the project and log on.

1. Click **Start** > **Programs** > **Micro Focus** > **StarTeam Cross-Platform Client <version>** .
2. Choose **Project** > **New** or **Project** > **Open** . The **Log On to [server_name]** dialog box opens.
3. Type the **User Name** and **Password** in the appropriate fields.

   Note: Passwords are case sensitive and may have length restrictions.
4. Check **Save As Default Credentials For This Server** to save your default credentials for this server configuration as this user name and password. This name will then appear in parentheses after the server configuration name in lists. The **New Project Wizard** or the **Open Project Wizard** displays allowing you to create a new project or work on an existing project. On the Microsoft Windows platform, logging on to StarTeam also displays the **Toolbar Utility** icon in your system tray. The **Toolbar Utility** caches a default credential for each StarTeam Server you access during a work session. This reduces the number of times you need to log on to the same StarTeam Server.
5. Optionally, click the **Toolbar Utility** icon and choose **Open** to see the list of servers for which you currently have default configurations.
6. Optionally, double-click the **Toolbar Utility** icon and choose **Use Same Name/Password** to access several server configurations with only one logon.

Logging on as a Different User

Before you can create a new project or open an existing project, you must select a server configuration for the project and log on.

1. Click **Start** > **Programs** > **Micro Focus** > **StarTeam Cross-Platform Client <version>** .
2. Choose **Project** > **New** or **Project** > **Open** .

3. Click the **Log On As** button. The **Log On to [server_name]** dialog box opens.
4. Type the **User Name** and **Password** in the appropriate fields.

   Note: Passwords are case sensitive and may have length restrictions.

5. Check **Save As Default Credentials For This Server** to save your default credentials for this server configuration as this user name and password. This name will then appear in parentheses after the server configuration name in lists.
6. Click **OK**.

Logging Off

This procedure shows you how to log off a StarTeam Server.

1. Switch the StarTeam client window to a project view on the StarTeam Server of which you want to log off.
2. Click **File** > **Log Off** . A confirmation dialog box appears asking if you want to completely log off this StarTeam Server.
3. Click **Yes** to continue. StarTeam closes all the views on this StarTeam Server that you have logged into during this StarTeam session.

Accessing Projects and Items

StarTeam can search or open URL shortcut links to projects, views, folders, and items (files, change requests, requirements, tasks, and topics). By creating shortcuts, you can easily and quickly access specific items in a project.

Search Use search to perform a server-wide search for specific text and file names.

Desktop Shortcuts If you will be accessing a project view frequently, you may want to save the view as a shortcut on your desktop. Double-clicking the shortcut both starts the application and opens the view associated with the shortcut. You can also create desktop shortcuts to items that you are tracking. Opening the shortcut starts the application, opens the project view in the configuration it had when the shortcut was created, and displays the item’s **Properties** dialog box.

URL Shortcuts You can create URLs and HTML representations for items and copy them to the Microsoft Windows **Clipboard**. Depending on the application, a paste operation transfers either the URL or HTML data to the application. For example, you can copy the names of a list of files to a Microsoft Excel spreadsheet using the HTML representation. You can email the URLs for a list of files to a coworker to use in StarTeam, as long as your email application does not convert the paste operation to HTML. Not all applications support pasting the HTML representation, although Word, Excel, and Outlook do support HTML data.

Like other URLs, StarTeam URLs include the name of the server for the connection. In some organizations, StarTeam servers may be reached from both the Internet and the corporate intranet. In such cases, a server may have two different IP addresses. If you configure the server list to reference a server by its IP address, rather than its DNS name, then any URLs generated by the client will work only from the network on which that IP address exists.

The type of URL that your StarTeam client creates for an item is set, per item, on their respective tabs in the **Personal Options** dialog box. There is one exception to this. Because there is no way to set the URL type for a folder, folders always use the ID-based URL type. Also from the **Personal Options** dialog box, you specify the templates used to create an item’s HTML representation. For projects, views, and folders there is no HTML representation.

- If the URL is a reference to a project, the default view of the project opens.
- If the URL is a reference to a view or folder, then the view or folder opens.
- If the URL is a reference to an item, the item's view opens, the item's parent folder is selected in the folder tree, the item type is selected, and the item itself is selected in the item list or tree on the upper pane.

The URL can be ID-based or name-based. ID-based is the initial default for each item type. The sample URLs below show the basic differences between ID-based and named-based URLs. They both represent a file in that root view of a project. The ID-based URL is the first of the two.

```
starteam://hostname:49201/12;ns=Project;scheme=id/
154;ns=View;scheme=id/869958;ns=File;scheme=id;scope=full;
starteam://hostname:49201/myproject/myview/path to myfile;scope
full;
```

The advantage of using the ID-based URL is that an item can be moved to a different folder (or a file’s name can be changed) and the item can still be located. The advantage of a name-based URL is that the URL can resolve to different StarTeam objects at different points in time. For example, if a file is deleted and then added again (with a new ID), it can still be located.

Search

As long as the StarTeam Server has search enabled, you can use your client to search for text. Note the following:

- StarTeam will search all projects, and all server configurations, as specified by the StarTeam Server. This includes the asset file name.
- Attachments for change requests, requirements, tasks, and topics will be searched. This includes the file name of the attachment.
- Text capitalization is ignored during search.
- Refer to the *StarTeam Server Administration Tool Help* to learn how to enable and configure search on the server.

To search for text:

1. Open a project.

   Note: You must have a project open to enable search.
2. Enter search text in the text field in the top right of your client and click **Enter**. The search view opens with your results. The results include the server and project in which the result was found.
3. There are several options for filtering:
   - Select the servers to search.
   - Select a project.
   - Select the buttons of an asset type.
4. Click the result's **Open** button (➡) to open the item. For the StarTeam Web Client, click the **Goto** button.

Wildcard Prefix Queries

The following symbols can be used for wildcard searches:

? Searches for single character wildcard. The single character wildcard search looks for terms that match that with the single character replaced. For example, to search for "text" or "test" you can use the search `te?t`.

***** Searches for multiple character wildcard.

Multiple character wildcard searches looks for 0 or more characters. For example, to search for database, datamart, datahub or dataset, you can use the search `Data*`.

You can also use the wildcard searches in the middle of a term. For example: `te*t`.

Note: You cannot use an `*` or a ? symbol as the first character of a search

Search Customization

You can use custom operators and StarTeam type/field names to enhance your search.

Example Search Phrase	Results
`hello world`	Searches for items containing `hello` or `world`, preferably both.
`hello -world`	Searches for items containing `hello`, but not `world`.
`hello +world`	Searches for items that must contain `world` and preferably contain `hello`.
`"hello world"`	Searches for the exact phrase in quotes.
`hello AND world`	Searches for items that must contain both `world` and `hello`.
`hello OR world`	Searches for items containing `hello` or `world`.
`hello NOT world`	Searches for items containing `hello` but not `world`.
`Content: hello`	Searches for file items containing `hello` only in its content.
`Attachment:hello`	Searches for items containing `hello` only in one of its attachment's content.
`(world OR universe) AND hello`	Group clause search for items containing either `world` or `universe` and must contain `hello`.
`Synopsis:(+test +"hello world")`	Searches for item synopsis fields with text that contains both the word `test` and the phrase `hello world`.
`hello type:cr`	Searches for items which contain `hello` or are of **type** `cr`, preferably both. Non-change requests can still show up if they are better matches. A *type search* searches for formal StarTeam type names, ignoring capitalization: *Topic*, *Concept*, *File*, *Story*, *Requirement*, *Change request* or *CR*, *Task*, *Whiteboard*, *Folder*, or *Sprint*.
`title:"my item title"`	Search the **title** field for the exact text `my item title`. You can search by any field using the format `field:value`, ignoring capitalization, lowercase by convention, no spaces around the colon. Some examples of field names: `name`, `title`, `content`, `synopsis`, `owner`, `description`, `comments`, and `note`. **Note:** Building a search query using user-defined custom fields, `ID` or `Enumeration` properties need to use special prefixes when referring to the property name. See the next section for more information.

Property Display Name Search

Field parameter accepts property's display name.

If the display name has a space, use quotes. For example:

`"Test Command": apache + type:ChangeRequest`

`"CR Number":1`

`"Task Name":tasktop`

`+Status:Pending +type:Task` returns all pending tasks.

Field Name Format for ID, Enumeration, and Custom Fields

When searching for text using a field name, certain StarTeam field types require special consideration. Enumerated fields and ID fields require the following preface before the field name: `Str_`. For example:

```
str_addressedby: "Joe Smith"
str_addressedby: Joe Smith
str_addressedby: Joe
```

The second example will interpret `Smith` as a separate `OR` match and will return all results with "joe" or "smith".

StarTeam's user-defined custom fields require a `Usr_` prefix. For example:

- A custom enumerated field named **Color**: `Str_Usr_Color:Red`.
- A custom user ID field named **StartedBy**: `Str_Usr_StartedBy:"Joe Smith"` .

Field Names

The following fields are available for each of the types as listed below.

All Item Types

The following fields are used for all types, except as noted below.

- `Comment`
- `Description` (except for Plan, Task, and Whiteboard)
- `DotNotation`
- `Str_CreatedUserID`
- `Str_DeletedUserID`
- `Str_ModifiedUserID`

ChangeRequest

- `Category`
- `Component`
- `ChangeNumber`
- `ExternalReference`
- `Fix`
- `Str_AddressedBy`
- `Str_AddressedInView`
- `Str_EnteredBy`
- `Str_Platform`
- `Str_priority`
- `Str_Responisbility`
- `Str_Severity`
- `Str_Status`
- `Str_StreamState`
- `Str_Type`

	StreamState Synopsis TestCommand WorkAround
Concept	ConceptName Content DiagramSource
File	ArchivePath Compression HiveArchiveFormat LocalName LocalPath MD5 Name Path ProjectPVCSRev StArchiveName SymbolicLinkTarget
Folder	CreateProject ExcludeSpec LocalPath Name Path Working
Plan	PlanName
Requirement	AmbiguitiesFound ChildType Comments DiagramSource ExpectedEffort ExternalReference ExternalTestProjectURL ExternalTestURL HighEffort LowEffort Notes Owner ParentRequirementName Path Rank RequirementNumber RevisedDescription RichDescription Str_MosCow Str_Owner

	• Str_Priority • Str_ResponsibileIDs • Str_ReviewedByIDs • Str_Status • Str_StreamState • Str_Tag • Str_Type • Str_UDAGrouping • StreamState
Sprint	• SprintName • Str_NotificationIDs • Str_Owner • Str_TargetReleases • Str_UserIDs • Team
Story	• AcceptanceCriteria • ExternalTestProjectURL • ExternalTestURL • Notes • RichDescription • Sponsor • Sprint • StoryOrder • StoryParentName • Str_IncomingState • Str_MosCow • Str_NotificationIDs • Str_Risk • Str_Sponsor • Str_StreamState • Str_StoryType • Str_StoryValue • Str_Tag • Str_TargetReleases • Team
Task	• RichDescription • Path • Story • Str_StTaskResourceIDs • Str_StWorkUserID • Str_TaskResponsibility • StTaskAttentionNotes • StTaskDescription • StTaskGU • StTaskParent • StTaskMSProjectFileName • StTaskName

	• `StTaskNotes` • `StTaskNumber` • `StTaskOrder` • `StTaskUnique` • `StTaskWBSCode` • `Team`
Test	• `Path` • `TestParentName`
Topic	• `ParentTopicTitle` • `Path` • `RecipientNames` • `Str_Priority` • `Str_RecipientIDs` • `Str_Status` • `TopicNumber`
Whiteboard	• `Color` • `WhiteboardName`

Copying and Opening URLs

The StarTeam client can open URL shortcut links to projects, views, folders, Not-in-View folders, and items. This enables you to quickly access specific locations in a project. You can also copy a URL to the clipboard, which allows you to easily move its contents to an appropriate application, such as an email client or a document.

Like other URLs, StarTeam URLs include the name of the server for the connection. In some organizations, StarTeam servers may be reached from both the Internet and the corporate intranet. In such cases, a server may have two different IP addresses. If you configure the server list to reference a server by its IP address, rather than its DNS name, then any URLs generated by the client will work only from the network on which that IP address exists.

Note: StarTeam shortcuts use the file extension `.stx`. For shortcuts to saved View Compare/Merge sessions, it uses `.vcmx` and `.vcms`.

Tip: Double-click a StarTeam URL link in an email or other application to open StarTeam to the specified project, view, or item. In **Windows Explorer**, StarTeam links display a small StarTeam icon which you can double-click to open.

Copying a URL to the Clipboard

1. Select one or more items in the upper pane.
2. Right-click the selected items and choose **Copy URL to clipboard**. This action places in the clipboard a plain text version of the URL to the selected items and an HTML representation of the links to the selected items. From the clipboard, you can paste the URL to a selected application.
3. Paste the URL to the application of choice.

   Note: Copying a URL to the clipboard is equivalent to dragging an item or items from the list pane or folder tree onto an application. Not all applications support pasting the HTML representation, although Word, Excel, and Outlook do support HTML data.

Opening a URL

1. Choose **Project** > **Open StarTeam URL**.
2. Type a valid URL to a project, view, folder, or item in the **Open StarTeam URL** dialog box. For example: `starteam:/hostname:49201/myproject`.
   - If the URL is a reference to a project, the default view of the project opens.
   - If the URL is a reference to a view or folder, then the view or folder opens.
   - If the URL is a reference to an item, the item's view opens, the item's parent folder is selected in the folder tree, the item type is selected, and the item itself is selected in the item list or tree on the upper pane.
3. Optional: Clear the **Open URL in new window** check box if you want the view to open in the same window you have open.

   Note: On the **Workspace** tab in **Tools** > **Personal Options**, you can define the default setting for the way a view displays in your workspace when you open a link to a StarTeam URL.
4. Click **OK**.

Creating Shortcuts

For easy access to items that you are tracking, you can save shortcuts to them on your desktop. Opening a shortcut starts the application, opens the project view in the configuration it had when the shortcut was created, and displays the **Properties** dialog box for the item.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click the appropriate component tab.
3. Right-click an item in the upper pane and choose **Save Shortcut**.

   This opens the **Save As** dialog box.
4. Type a name for the shortcut in the **File Name** field if you do not want to use the default name.

   Be sure to keep the `.stx` extension.
5. Select a location, usually your desktop, for storing the shortcut and click **Save**.

Note: To open the item from the shortcut you saved, double-click the shortcut file (`.stx`).

Opening StarTeam Agile

You can quickly and easily launch StarTeam Agile from within the Cross-Platform Client.

Your server administrator must first specify the `StarTeamAgileWebAddress` option information in the `starteam-server-confs.xml` file and then restart the server before you can use this feature.

1. Choose **Project** > **Open StarTeam Agile** from the main menu in the Cross-Platform Client. A new browser window opens with the StarTeam Agile Sign In page.
2. Sign in to StarTeam Agile.

Configuring Your Client

This section contains topics related to configuring the StarTeam Cross-Platform Client.

Connecting to a Server Configuration

StarTeam stores all projects on the StarTeam Server, which may contain numerous server configurations. You can access one or more StarTeam Server from the application. However, if you have more than one server configuration running on the same computer, each server configuration must use a unique protocol and port combination. After the server is added, you can access whatever projects are available for the current server configuration of that StarTeam Server.

Managing StarTeam Server access includes adding, deleting, or modifying the server configuration properties. You can accomplish these tasks as part of creating or opening a project in one of the clients or from the **Server Administration** tool.

1. Do one of the following:
   - In your client, click **Project** > **New** or **Project** > **Open** .
   - On the StarTeam Server, click **Start** > **Programs** > **Micro Focus** > **StarTeam Server xxxx** > **StarTeam Server** . Alternatively, from the StarTeam Cross-Platform Client, you can select **Start** > **Programs** > **Micro Focus** > **StarTeam Cross-Platform Client xxxx** > **Server Administration** . These actions display the **Server Administration** tool.
2. Do one of the following:
   - Click **Add Server** in the **New Project** or **Open Project** wizard.
   - Click **Server** > **Add Server** in the **Server Administration** tool.
3. Type a unique, easy-to-remember description in the **Server description** field. It is not case-sensitive and may contain colons (:)
4. Type or browse for the computer name or IP address in the **Server address** field.

   Note: See your administrator for the server address, protocol, and endpoint information. Your administrator can also tell you what MPX profile to use if your server configuration uses StarTeamMPX.
5. Type the endpoint (TCP/IP port number) associated with the protocol in the **TCP/IP endpoint** field.
6. Choose to specify any of the following optional settings:

Compress transferred data	Check if you want to use compression.
Encryption type	Select to encrypt data transferred between your workstation and the StarTeam Server. Encryption protects files and other project information from being read by unauthorized parties over unsecured network lines. The encryption types are ordered (top to bottom) based on speed. Each type is slower, but safer, than the type that precedes it.
MPX Profiles	Click and choose a different profile if you are using StarTeamMPX on the client and do not want to use the default profile (usually **Unicast On-site**).

7. Optionally, click **Properties** to review the connection properties of the selected profile.
8. Click **OK**.

Changing the Default Language and Locale

You can change the default language and local for the application by modifying the `StarTeamCP.stjava` file in the root StarTeam directory.

1. Open the `StarTeamCP.stjava` text file in any text editor.
2. Edit the line `options=-Xmx256m` to be one of the following languages:

German	`options=-Xmx256m -Duser.language=de -Duser.country=DE`
English	`options=-Xmx256m -Duser.language=en -Duser.country=EN`
French	`options=-Xmx256m -Duser.language=fr -Duser.country=FR`
Japanese	`options=-Xmx256m -Duser.language=ja -Duser.country=JA`

Changing Your Password

Occasionally, you might need to change your password. For example, you might need to change it every three months to follow company policy.

1. Choose **Tools** > **My Account** to open the **My Account** dialog box.
2. Click the **Logon** tab.
3. Do one of the following:
   - Type your new password in the **Password** and **Confirm** fields.
   - Check **Set a Blank Password** if your administrator allows blank passwords and you want to use one.
4. Click **OK**.

Auto Client Update

Administrator's can distribute StarTeam Cross-Platform Client updates to each client. To run the update, choose **Download Client Update** from the **Help** menu.

Configuring an Alternate Editor, Merge, or Comparison Utility

You can specify an alternate editor, merge utility, and comparison utility to use in the application if you don't want to use the default tools for those functions.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Click the **File** tab.
3. Check one or more types of alternate applications you want to use: **Editor**, **Merge Utility**, and **Comparison Utility**.
4. Type or browse to the path for executable file for each selected application.
5. Type any options you want to use in the **Options** text boxes for the selected applications.
6. Click **OK**.

Note: For non-Microsoft Windows systems, specify a command to use for launching files in an alternate application in the **Open with...** text box on the **File** or **Folder** tab in the **Personal Options** dialog box.

Configuring the Display Order of Component Tabs in the Client

This procedure describes how to change the order component tabs display in the upper pane of the client.

Note: When reordering the tabs in the upper pane, do not make the **Audit** tab the first tab. Listing all the audit entries can take a very long time.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. On the **Workspace** tab, click the **Advanced** button at the bottom. The **Advanced** dialog box opens.
3. In the **Component Order for server [ServerName]**, click the **Component Order** button.

   This opens the **Select Component Order** dialog box which displays two lists of components. On the left, there are available components to add to the display, and on the right, the current components in the order they are being displayed.

4. Do one of the following:

Remove a component from current display	Select it on the right and click **Remove**.
Add a component to current display	Select it on the left and click **Add**. The component is added to the end of the list.
Change the order of component tabs	Remove them all from the list on the right, then add them back in the desired display order.

5. Click **OK**.

Note: You must close and reopen your project to see the changes.

Show Advanced Menus

The **Show Advanced Menus** option allows you to control whether standard or advanced menu items display in the component menu selected from the Cross-Platform Client main menu. Choose **Tools** > **Show Advanced Menus** from the main menu to turn the feature on or off. **Show Advanced Menus** is turned on by default and a check box displays when the option is turned on. When **Show Advanced Menus** is not on, you see a standard menu with the advanced items hidden.

The table below shows the menu items that are only available when **Show Advanced Menus** is on.

Component Menu	Advanced Menu Items Available
Project	• **New** • **Properties** • **Access Rights** • **Delete** • **Connection Properties**
View	• **New** • **Delete** • **Access Rights** • **Labels** • **Promotion**
Folder Tree	• **Labels** • **Advanced** (Includes: **Behavior**, **References**, and **Access Rights**) **Note:** You can select **View Compare/Merge** from the advanced and standard menus.
Item Menus and Items Pane context menus (File, Folder, Change Request, Requirement, Task, and Topic)	• **Advanced** (Includes: **Export**, **Behavior**, **Access Rights**, **Component Access Rights**, and **Customize**) **Note:** You can select **View Compare/Merge** from the advanced and standard menus. • **Reports** • **Charts** • **Linked Files** • **Copy** • **Paste** • **Labels** • **Flag**
Tools	**StarTeam Log**

Controlling How File Status Information is Stored

File status information about the files you are working on is stored on your workstation either in a central location or in a child folder (named `.sbas`) of each working folder.

You can set the file status property for a specific view. The view property defaults to the storage method that you selected as a personal option. When changed from that default, the view property take precedence over your personal option for the view.

Note: You can also set your **Personal Options** to control file status information for all your files, unless those files are in views for which you have set the view property for file status.

1. Choose **View** > **Properties** . The **View Properties** dialog box opens.
2. Click the **Name** tab.
3. Select the **Central** or **Per Folder** option button in the **Repository** group.

   The per-folder option is most useful in the special case where multiple users are sharing a working folder. For example, on a shared network drive.

   For example, suppose several users all check files in and out of a shared working folder. If these users have set the central repository option for file statuses, the statuses are stored on each of their computers. Whenever a user makes a change to a file in the working folder, the status for that file is undated only on that user's computer. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files. Using the per-folder option causes the statuses to be updated within the working folder itself. Everyone has access to those status changes and **Unknown** statuses do not occur.
4. Click **OK**.

Tip: Select **Default (Central)** to return to using the **Personal Options** settings.

Distribute starteam-client-options.xml through StarFlow Extensions

Check `starteam-client-options.xml` into StarFlow Extensions to ensure all clients use customized personal option values. The users need to have access right permissions to see StarFlow Extensions project and to see and check-out the files in order to use the functionality.

Note: Do not use options in the `starteam-client-options.xml` that are path specific.

Customizing Personal Options

Personal Options allow you to customize the application by adjusting the way the components work.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Select the tab which contains the options you want to change and make the changes.
3. Click **OK**.

Note: It is necessary to close and reopen the project for component tab order changes to take effect.

Customizing the Detail Pane

You can modify the display format and content of the **Detail** (lower) pane on a per-workstation basis by placing correctly named and formatted HTML templates in the same directory as the `starteam-servers.xml` and `starteam-client-options.xml` files. For example, on a Microsoft Windows system these files could be located in the `C:\Documents and Settings\USER\Application Data\Micro Focus\StarTeam` folder.

Sample **Detail** pane templates are installed under the root installation folder in the `samples\details-templates` folder.

The name of the template file controls the StarTeam component to be modified. For example, a template named `changerequest.details.html` controls the format and content of the **Detail** pane display for the change request component.

1. Create an HTML template file for the corresponding component that you wish to customize. For example, if you want to format the contents of the **Detail** pane for a change request, you would create a template file named `changerequest.details.html`.

   You must use the following file names for the component detail panes that you wish to customize:

   - `folder.details.html`
   - `file.details.html`
   - `changerequest.details.html`
   - `task.details.html`
   - `topic.details.html`
   - `requirement.details.html`
   - `changepackage.details.html`
2. Make any desired modifications to the template file.

   Follow the formatting example in the sample template file. The fields used in the **Detail** pane HTML templates are recognized by the client when they are contained between double tilde ~~ characters. For example: `~~Status~~` represents the Status field found in the **Change Request Properties** dialog box.
3. Save the template files in the same directory as the `starteam-servers.xml` and `starteam-client-options.xml` files. For example, on a Microsoft Windows system these files could be located in the `C:\Documents and Settings\USER\Application Data\Micro Focus\StarTeam` folder.

Displaying Additional Fields

You can change what column headers appear in the upper pane or in the **History** tab. These headers can be hidden, displayed, or re-ordered, and in the upper pane, affect what you can find using the `Find` command.

1. Choose **Filters** > **Show Fields** from the appropriate item menu or context menu. For example, if files are displayed in the upper pane, choose this option from the **File** menu.
2. The **Show Fields** dialog box displays two lists. The **Available Fields** list contains all the fields that could be displayed as column headers but are not currently displayed. The **Show These Fields in This Order** list displays all the fields that are currently displayed.

   Do any combination of the following:

Display additional fields	Select the fields to display as the column headers from the **Available fields** list. Then click **Add**.
Stop displaying fields	Select the fields to be removed from the **Show these fields in this order** list. Then click **Remove**.
Change the order of the fields	Drag each field name to the desired location in the **Show these fields in this order** list.
3. Click **OK**.

Tip: Double-clicking a field name moves it from one list box to the other. The **Show Fields** dialog box initially displays the most commonly used fields. Check the **Show Advanced Fields** check box to select from a complete list of the available fields.

Displaying and Customizing Logging Options

The `StarTeam.Log` file records the operations performed on your workstation during a work session. Reviewing logs can help you or your administrator troubleshoot errors or failed operations.

`StarTeam.log` contains data about operations sent from your workstation to one or more servers, depending on what project views you have open. This data includes the name of the project so that you can isolate data for a particular server, when necessary. Depending upon the selections made in **Personal Options** dialog box, your `StarTeam.Log` file can record the following types of information: error messages, operation summaries, and details about the individual commands required to perform each operation.

To view the contents of the `StarTeam.Log` file, do one of the following

- Choose **Tools** > **StarTeam Log** in the StarTeam client.
- Import and view the data from a `StarTeam.Log` file using any application that supports tab-delimited fields. For example, if you save the file with a `.csv` extension, you can open the file in Microsoft Excel.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Type or browse for the location of the `StarTeam.Log` file in the **Log Output Path** field.

   The default is the location in which the application is installed. For example, `C:\Program Files \Micro Focus\StarTeam client_name\Log`.

   The current log file is always named `StarTeam.log`. Log files from earlier sessions of the application include the date and time the file was last modified.
3. Select the types of data you want to include in `StarTeam.Log`.
4. Click **OK**.

Displaying Notifications in the StarTeam Cross-Platform Client

Some personal options control how often you are alerted about new items using **System Tray** notification. In the clients, notification icons appear on the **Status Bar**.

While you are running the application, you can check for changes in items that may affect you. This feature does not apply to files and audit entries. The application notifies you when:

- A change request, requirement, or task becomes your responsibility or a topic names you as a recipient. If a topic has no recipients listed, no one receives notification.
- A requirement or task that is your responsibility or a topic for which you are a recipient has changed.

Note: The defect, requirement, task, or topic icon display at the right end of the **Status Bar**.

To enable System Tray notification

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Select the **Change Request**, **Requirement**, **Topic**, or **Task** tab.

   Note: If you have StarTeam Agile installed and configured to point to an existing StarTeam Server, you will also see **Sprint** and **Story** tabs.
3. Check **Check for New or Modified [item]** in the **System Tray Notifications** group box.
4. Type the number of minutes for the time interval between checks for items that need your attention. The default is 10.

The dialog box displays the icon that will appear in the **System Tray** for that particular item.

5. Click **OK**.

To open a System Tray notification item

1. Double-click the defect, requirement, task, or topic notification icon in the **System Tray** to open the **[Project] New [Item Type]** dialog box which lists the items that need your attention.
2. Double-click the item to display its properties.

   You might want to take notes while displaying the item properties because as soon as you close the dialog box the item disappears.

   Note: If your administrator has enabled email notification, you will automatically receive email messages notifying you about change requests for which you are responsible, about changes in any requirements and tasks for which you are responsible, and about changes in any topics for which you are a recipient. Email notification is client independent and you do not need to run the application to receive notifications. You can, however, use the **System Tray** notification with or without email notification.

Editing Your Account Information

Occasionally, you must update information about your user account information on the server, such as your name, contact information, and password.

1. Choose **Tools** > **My Account** to open the **My Account** dialog box.
2. On the **General** tab, enter any missing information or change any incorrect information in the fields available.
3. Click **OK**.

User-Defined Property Fields

StarTeam Enterprise Advantage and Enterprise licenses enable you to customize the repository by adding any number of user-defined property fields to files, change requests, requirements, topics, and tasks.

You can also change the properties of some existing items. For example, you can customize the **Priority** change request field. Instead of using the default values (`Yes` and `No`), your company can change the item properties so that StarTeam prioritizes change requests on a scale from 1 (high) to 10 (low).

The application adds new property fields and modified property fields to the database used by your current server configuration. These fields are available in the same locations as other item property fields.

Note: When you customize the database, you should use the `stcmd` command line to exclusively lock the StarTeam Server.

Modifications to Existing Fields

You can modify some existing property fields and any custom fields that have been created. The property fields that can be modified are always enumerated types.

For example, **Priority** is an existing change request property that can be customized. On the **Synopsis** tab of the **Change Request Properties** dialog box, **Priority** has the values `Yes` and `No`, but is implemented as an enumerated type with `Yes` having the numeric code of `1` and `No` having the numeric code of `0`. If your company prefers to prioritize change requests on a scale from 1 (high) to 10 (low), you can add additional values. You can use the current numeric codes 0 and 1 with new names or disable them.

Tip: Try to take advantage of the existing codes in some way.

For example, you might change the name of the enumerated value `No` to `Not Prioritized`. Then you can query for `Priority equal to 0` to identify the change requests that need to be prioritized.

You might also change the name `Yes` to `Priority 1`, as it already has the code `1`. Then you can add additional values for `Priority 2` through `Priority 10`. Any numeric codes that you assign must have a value greater than 100, because the application reserves the values 0 through 100 for internal uses. So you might add a value with the name `Priority 2` and the numeric code `101`, `Priority 3` with the numeric code `102`, and so on.

Tip: Queries use the numeric codes to make comparisons based on enumerated values. Using the above codes, you can write a query to locate Priorities 1, 2, and 3, by querying for `Priority greater than 0 and less than 103`.

By default, sort and list boxes display items in the order that they appear in the **Add Field** or **Modify Field** dialog boxes. You can use the sort options to sort by name in ASCII or alphabetical order.

Translation of Display Names and Enumerated Values

When you create a custom field, you provide a display name for that field. If you create the field as an enumerated type, you must also provide names for the enumerated values. The client also displays all of these names to users. The names that you specify when you create the field become the default names for that field.

If you do not set up translations for the names of a custom field, users will always see the default names. However, if you translate the names associated with a field, users will see the names provided for the locale specified for the operating system of that client.

For example, suppose you have a custom field for which you specified all the default names in English. Then you translate the names (a display name and several names for enumerated values) into French. Users whose clients use French operating systems will see the French names. All other users see the default names.

In more complicated examples, you need to understand how locales and languages are defined. Each locale or language is defined by using at most three parts separated by hyphens. These parts represent the language itself, the country it is spoken in, and a variant of that language. Most languages are represented by just the first two of these parts.

For example, Dutch has the following three entries in the language table:

- Dutch [nl]
- Dutch (Belgium) [nl-BE]
- Dutch (Netherlands) [nl-NL]

The bracketed section represents the locale or language. In this case, [nl] represents a generic Dutch, while [nl-BE] and [nl-NL] represent Dutch as it is spoken and written in Belgium and in the Netherlands, respectively.

If you translate the names from a custom field into generic Dutch, all Dutch operating systems can use those names. If you translate the names into both generic Dutch and the Dutch used in Belgium, the client treats the names as follows:

- Users whose clients run on operating systems with the locale set to the Dutch of the Netherlands (the [nl-NL] entry) see the generic Dutch translations. If an enumerated name is not translated into Dutch, these users see the default name for that value.
- Users whose clients run on operating systems with the locale set to the Dutch of Belgium see the translations for the Dutch of Belgium.

If an enumerated name is not translated into the Dutch of Belgium, they see the generic Dutch name for that value. If a name is not translated into either the Dutch of Belgium or generic Dutch, they see the default name for that value.

Accordingly, if you are generally satisfied with the generic Dutch translations, but want to use a different spelling or a different word for just one value in the Dutch of Belgium, you can translate the name of that value into the Dutch of Belgium.

Note: The Java VM reads the locale information setting for the client's operating system when it starts running. If you change that setting, you must restart the Java VM.

Tip: Generally, an organization standardizes on one language for the default values of all custom fields. Otherwise, the application allows you to use a different default language for every custom field. For example, when you create a custom field using English names, English names become the default for that field. If someone else creates a custom field using Japanese names, the default names for that field are Japanese, unless they are changed later.

Translating Display Names

When you create a custom field, you need to provide a display name for that field.

1. Do one of the following:
   - While creating a custom field with the **Add Field** dialog box, click **Translate**.
   - Display the properties of a custom field. Click **Advanced** > **Customize** from the **File** or **Change Request** menus. Select the field to be modified. Click **Edit**, to display the default values in the **Modify Field** dialog box. These values may not be what you see in your client. For example, the defaults may be in English, and you may be using the StarTeam Cross-Platform Client with a French locale. Click **Translate**. The **Translate Field** dialog box appears.
2. Select the language you want in the **Language** list. If it does not appear, click **Add**.
   a) In the **Add Language** dialog box, select a language from the **Language** list.
   b) In the **User Defined** field, type an existing locale name, such as `fr-FR` to quickly select the language by its locale name.
   c) Type a new locale name.

      A user-defined locale name must follow the formatting rules for locale names. You can use up to 8 characters: letters, numbers, and one or two hyphens to separate the locale name into two or three parts. Underscores display as hyphens. (No spaces are allowed.) Regardless of the case you type, the first part will always be lowercase and the second part will always be uppercase. After its creation, the user-defined language becomes a member of the language list for the current field. It is not available for other fields unless you also create it for those fields. To see the values in a user-defined language, users must set their workstations to that locale.
   d) Click **OK**.

   Be aware that adding a language to one field does not automatically add it to the language list for any other field. Also, if you do not translate any values, the language will disappear from the language list for this field.
3. Optionally, type the display name for the field in the new language in the **Translated Display Name** field.
4. Optionally, translate enumerated values.
5. Click **OK** to return to the **Add Field** or **Modify Field** dialog box.
6. Click **OK** to return to the **Customize** dialog box.
7. Click **Close.**

Translating Enumerated Values

If you create an enumerated type field, you must also provide names for the enumerated values. All of these names are displayed to users in the client.

1. Select an enumerated value to be translated.
2. Click **Edit**. The **Translate Enumerated Value** dialog box displays.

3. Type the translated value in the **Translated Text** field.
4. Click **OK** to return to the **Translate Field** dialog box.
5. Repeat steps 1-4 for other enumerated values.

Displaying Custom Property Fields

It is helpful to be able to look at a list of all custom property fields that have been created for a certain type of item. Also, having created such fields, you may wish to select them for display on the upper pane of the project view window.

Displaying Custom Property Fields for an Item on the Properties Dialog Box

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Select a component tab. This action displays items in the upper pane.
3. Highlight an item on the upper pane, and select **Properties** from the component menu or context menu. These actions display the **Properties** dialog box.
4. Click the **Custom** tab to see the custom fields created for the selected item.
5. Click **OK**.

Displaying Custom Property Fields in the Upper Pane

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Select a component tab. This action displays items in the upper pane.
3. Select **Filters** > **Show Fields** from the component menu or context menu. The **Show Fields** dialog box appears.
4. Select the fields you want to display from the **Available fields** list, and click **Add**.
5. Arrange the fields in the **Show these fields in this order** list by dragging and dropping the field names.
6. Click **OK** when you are finished. The fields you have selected appear in the upper pane.

Note: Only 60 columns can be displayed in the upper pane at a time. If the addition of custom fields raises the total over 60 fields, you cannot display all fields for an item as columns in the upper pane. Remember that databases have limitations as well, which can limit the number of custom fields you create.

Creating Boolean Fields

Boolean property fields are used to define true/false property values.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII

characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: `= \\.^$@,;!:#*&<>?-//%|[](())+"`

Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.

5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Boolean** from the **Type** list.
7. In the **Possible Values** list, `True` and `False` are listed by default. If you want to change the names of these values (for example to `Yes` or `No`), select the value and click **Edit**.
8. Optionally: Click **Add** to enter a new value. The **Add Value** dialog box opens. StarTeam reserves the numeric codes from 0 to 100, so this dialog shows numbers starting with 101.
9. In the **Default Value** box, select the value you want to be the default value for all items that have this field type as a property. If you want the default value selected to always be true, select the **Default is True** check box.
10. Verify that the values you have selected are the ones that you want. They cannot be changed once you exit the dialog box.
11. Click **OK**.

Creating Content Fields

Content property type fields can be used to store rich text values, such as HTML MIME types and the StarTeam Cross-Platform Client provides a rich text editor for users to edit them. Use the **Text** property type field to store standard text with no markup.

Note: A default value cannot be set for a this field because they normally contain unique data.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
   - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
   - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
   - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: `= \\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Content** from the **Type** list.

   Note: The default maximum length is 255 characters and cannot be changed.
7. Click **OK**.

Creating Date and Time Fields

There are two date and time property types that can be used to create new date or date and time fields. The **Date** type has only a date component, while the **Date/Time** type has both a date and a time component.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: = `\\.^$@,;!:#*&<>?-//%|[](())+"`

    Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Date** or **Date/Time** from the **Type** list. The **Date** type stores date values without requiring inclusion of **Time**.

    Optionally, select the **Default value** check box and type in a default date or time, using the appropriate format for your locale. This value automatically becomes the value used for all existing items that have the field as a property. It also becomes the default value for newly created items. You can change this value manually on the **Custom** tab of the item **Property** dialog box.

    Note: If you do not set a default, no value is placed in the database for any item with this field, unless you set the field to a value manually.
7. Click **OK**.

Creating Enumerated Fields

The order in which enumerated values appear in list boxes in the **Query** and other dialog boxes is the order in which they appear in the **Add Field** or **Modify Field** dialog box. This can be a code order or even alphabetic order, but only if they appear in that way in the dialog box. You can use drag-and-drop to rearrange the values.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.

3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: `= \\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.

5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Enumerated** from the **Type** list.
7. Click **Add** to enter the first value. The **Add Value** dialog box opens. StarTeam reserves the numeric codes from 0 to 100, so this dialog shows numbers starting with 101.
8. Use the displayed code or type another in the **Code** text box.
9. Type the name for this enumerated value in the **Name** text box and click **OK**.
10. Use drag-and-drop to arrange the values in the **Possible Values** list box.
11. Select **Supports multi-select** to provide users with the ability to select more than one value.
12. In the **Default Value** box select one or multiple enumerated values as the default for users to select. This value automatically becomes the value used for all existing items that have the field as a property. It also becomes the default value for newly created items. You can change this value manually on the **Custom** tab of the item **Property** dialog box.

    Many users create a value of **<none>** to use as the default, to indicate that no value is really being set. They also place this value first in the enumerated list, so that it displays at the top or bottom of the upper pane when the column for the field is sorted. If you are entering a date or time, use the appropriate format for your locale.
13. *Optionally*: After you have added the values, you can arrange them hierarchically and sort the values in parent/child relationships as desired.
    a) In the **Possible Values** list, select a code and click **Edit**.

       If you are adding a new value, click **Add**.

    b) Select **Make child of** and choose a value in the list to arrange the values in parent/child arrangements.

       Tip: Alternatively, you can also drag and drop the values in the list to arrange them hierarchically. When dragging a node, it becomes a sibling of the node it is dropped onto, and it is placed immediately after that node.

    c) Click **OK**.
14. Verify that the codes you have selected are the ones that you want. They cannot be changed once you exit the dialog box and click **OK**.

Creating Group and Group List Fields

Group fields can be used to store the user group values defined on the server. There are two group property types that can be used to create new group fields. The **Group** type allows users to only select one group, while the **Group List** is a multi-select option that allows multiple group values to be selected.

Note: A default value cannot be established for a group field, because if the **Group** is deleted, the value becomes invalid.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:

    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: = `\\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Group** (for a single group value selection) or **Group List** (for multiple group values selection) from the **Type** list.
7. Select the **Input required** check box to make this a required field. Required fields are rarely used with files, because they affect the **File Properties** dialog box only.
8. Click **OK**.

Creating Map Fields

Map fields can be used to store structured values on items where the value types are not required to be the same or exist on every item instance.

Note: A default value cannot be set for a this field because they normally contain unique data.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: = `\\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Map** from the **Type** list.

   Note: The default maximum length is 255 characters and cannot be changed.
7. Click **OK**.

Creating Rich Content Fields

A default value cannot be set for a rich content field because most rich content fields contain unique data.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: = `\\.^$@,;!:#*&<>?-//%|[](())+"`

    Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. From the **Add Field** dialog box, select **Rich Content** from the **Type** field.
7. Click **OK**.

Creating Text Fields

Text property type fields store standard text with no markup. Use the Content property type to store rich text values such as HTML.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
    - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
    - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
    - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: = `\\.^$@,;!:#*&<>?-//%|[](())+"`

    Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Text** from the **Type** list.

7. Use the default maximum length (255 characters) or type a number of characters from 2 to 20,000 in the **Length** text box.
8. Verify the **Length** is adequate, because you cannot change it after exiting the dialog.
9. Select the **Input required** check box if you want to make the field required. Required fields are rarely used with files, because they affect the **File Properties** dialog box only.
10. Click **OK**.

Creating Time Span Fields

Time span property fields can be added to specify a duration of time in days, hours, minutes, and seconds.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:
   - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
   - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
   - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.
3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: `= \\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **Time span** from the **Type** list.

   Optionally, select the default value **Time Span Days** check box and type in a default value for the number of days and/or select a default time (H = hours, M = minutes, S = seconds). This value automatically becomes the value used for all existing items that have the field as a property. It also becomes the default value for newly created items. You can change this value manually on the **Custom** tab of the item **Property** dialog box.

   Note: If you do not set a default, no value is placed in the database for any item with this field, unless you set the field to a value manually.
7. Click **OK**.

Creating User and User List Fields

User fields can be used to store the user values defined on the server. There are two user property types that can be used to create new user fields. The **User** type allows for the selection of only one user, while the **User List** is a multi-select option that allows multiple user values to be selected.

Note: A default value cannot be established for a this field because if the data is deleted, the value becomes invalid.

1. Select **[Component]** > **Advanced** > **Customize**. **[Component]** corresponds to the component for which you are creating a property field. The **Customize** dialog box opens.
2. Review the **Customize** dialog box to find the fields that can be customized. The fields and icon graphics accompanying them are as follows:

  - A field with a pencil in the center of the icon is an application field. It is always an enumerated type and is fully customizable. You can add, disable, rename, and reorder its values.
  - A new field icon displays the head and shoulders of a person. It can be one of several types and is fully customizable. If the new field is disabled, the icon is greyed-out.
  - An application field with an icon containing a small yellow lock in the lower left corner is a restricted enumerated type. You can change only the names the application displays for the values of this field.

3. Click **Add**. The **Add Field** dialog box opens.
4. Type the name for the new field to be used by the database in the **Field name** field. Use only alphanumeric characters and no spaces in this name. The name should be less than 31 ASCII characters (including `Usr_`) and not contain the following characters, which are not accepted by one or more of the databases that the application supports: `= \\.^$@,;!:#*&<>?-//%|[](())+"`

   Note: Be careful about selecting the field name, as it cannot be changed once you click **OK**.
5. Type the name that the application will display to users in the **Display Name** field.
6. Select **User** (for a single user value selection) or **User List** (for multiple user values selection) from the **Type** list.
7. Select the **Input required** check box to make this a required field. Required fields are rarely used with files, because they affect the **File Properties** dialog box only.
8. Click **OK**.

Customizing the Detail Pane

You can modify the display format and content of the **Detail** (lower) pane on a per-workstation basis by placing correctly named and formatted HTML templates in the same directory as the `starteam-servers.xml` and `starteam-client-options.xml` files. For example, on a Microsoft Windows system these files could be located in the `C:\Documents and Settings\USER\Application Data\Micro Focus\StarTeam` folder.

Sample **Detail** pane templates are installed under the root installation folder in the `samples\details-templates` folder.

The name of the template file controls the StarTeam component to be modified. For example, a template named `changerequest.details.html` controls the format and content of the **Detail** pane display for the change request component.

1. Create an HTML template file for the corresponding component that you wish to customize. For example, if you want to format the contents of the **Detail** pane for a change request, you would create a template file named `changerequest.details.html`.

   You must use the following file names for the component detail panes that you wish to customize:

   - `folder.details.html`
   - `file.details.html`
   - `changerequest.details.html`
   - `task.details.html`
   - `topic.details.html`
   - `requirement.details.html`
   - `changepackage.details.html`
2. Make any desired modifications to the template file.

   Follow the formatting example in the sample template file. The fields used in the **Detail** pane HTML templates are recognized by the client when they are contained between double tilde ~~ characters. For example: `~~Status~~` represents the Status field found in the **Change Request Properties** dialog box.

3. Save the template files in the same directory as the `starteam-servers.xml` and `starteam-client-options.xml` files. For example, on a Microsoft Windows system these files could be located in the `C:\Documents and Settings\USER\Application Data\Micro Focus\StarTeam` folder.

Configuring Projects to Use Alternate Property Editors

If your company has licensed StarTeam Enterprise Advantage, you can use alternate property editors (APEs). APEs are Java forms created specifically for your company to support a corporate process. Workflow processes are created for use with the forms. APEs can be created for the File, Change Request, Requirement, Task, and Topic components.

APEs use the StarTeam SDK to access a StarTeam server configuration. APEs can be customized because they are implemented in standard programming languages. Sample property editors for several StarTeam components are included with StarTeam Enterprise Advantage.

By default, every project uses the standard property dialogs. When an APE is ready for use, you must change the project properties so that all the views in a project will use the APEs. Once set, every view in the project must use an APE instead of the standard dialog. However, each view can use a different APE that you have checked into the appropriate view-specific subfolder in the Projects folder of StarFlow Extensions project.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Editors** tab.
3. In **Type**, select the component for which you want to create an APE.
4. Click **Browse** to navigate and find the correct APE. Generally, you enter the word `Locator` followed by the name of the APE. `Locator` is a program on each client workstation that distributes code and XML updates to client workstations.

   When `Locator` is specified and StarTeam requests the APE, `Locator` looks for the StarTeam Extensions project managed by the server configuration. If the project exists and contains an APE with the specified name, Locator copies the tip revision of that APE and its related files to the client workstation (if they are not already there). Then the APE will be used instead of the StarTeam standard dialog for that type of item.

   Caution: Take care when setting the APE for a project in production because the new setting takes effect immediately. It is important to test any changes in an APE prior to making this editor available to a wide audience.
5. Click **Add**, **Update**, or **Delete** depending on which action you want to perform to configure the project.
6. Click **OK**.

Activating Forms

After you save the form in an appropriate project directory in the StarFlow Extensions project, follow the instructions below to activate the form for use in the Cross-Platform Client.

Note: If you have alternate property editors (APEs) designated for use in the **Project Properties** dialog box, you must disable them to activate the customized Layout Designer form.

1. Launch the Cross-Platform Client and access the project or view in which the form will be used.
2. Select the appropriate item tab in the upper pane. For example, if you have modified the default **Change Request** form with Layout Designer, select the **Change Request** tab.
3. Choose **[Component]** > **Properties** to open the modified properties dialog. **[Component]** corresponds to the component for which you edited the form, for example a **Change Request**. Your modified properties dialog displays. To make further updates to the form, you can reopen the form in Layout Designer.

Projects

A project is a way to group related items (such as files and change requests) hierarchically. Views and folders enable you to organize these related items more efficiently. For example, if you create a project for a software product, the files containing the product's functional specification, marketing requirements document, source code, and test suites can each be stored in separate folders.

Views can be used in a variety of ways. For example, different views can be used so that developers see only the project's source code folder and its child folders, marketing personnel see only the project's marketing folder and its child folders, and so on. In this case, each view has a different folder as its root. Views also support branching and parallel development

At the view level or item by item, you can branch data such as files and change requests. The branching enables you to create a special variation of your product. For example, you can start on the 2.0 version of your product without hampering the creation of service packs for the 1.0 version

You can create a project on any StarTeam Server configuration, if your computer has access to that server and you have been granted the rights needed to create projects in that location. When you create a project, you must provide a project name and designate a working folder location for the project's root (initial) folder. The initial view of the project is created at the same time you create the project. It has the same name as the project, although you can change the name later if you choose. The root folder is also created at this time. If your computer is not currently set up to access the server on which the project will reside, you can add access to that server as part of creating the project.

To access the **Project Properties** dialog box, choose **Project** > **Properties** .

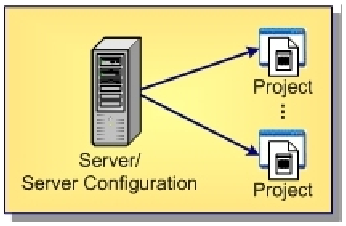
⚠ **Important:** The StarTeam Server creates new projects with only the **File** type pre-selected as a default for new views. Users can still change the project properties after the project is created, and they can change the item types included for any given new view. However, if the user changes nothing, by default new views will only include files when they are created. Adding other item types to the **Project Properties** (after the view is created) will NOT populate the items that were contained in the parent view (but left out during New View creation). If the user wants to bring the previous items into the new view, they must retrieve them by Rebasing from the parent view.

Project Structure

An instance of the StarTeam Server controls the storage of your files. Each StarTeam Server instance runs a server configuration. Below are some diagrams illustrating how all these pieces fit and work together.

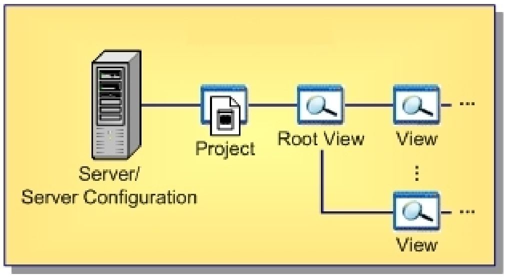
Server-level Hierarchy

The server can manage any number of projects.



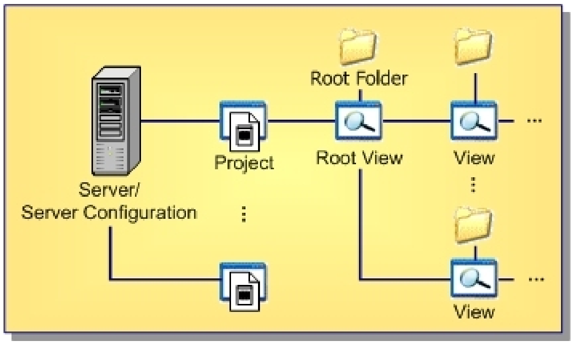
Project-level Hierarchy

Each project has one root view and any number of child views.



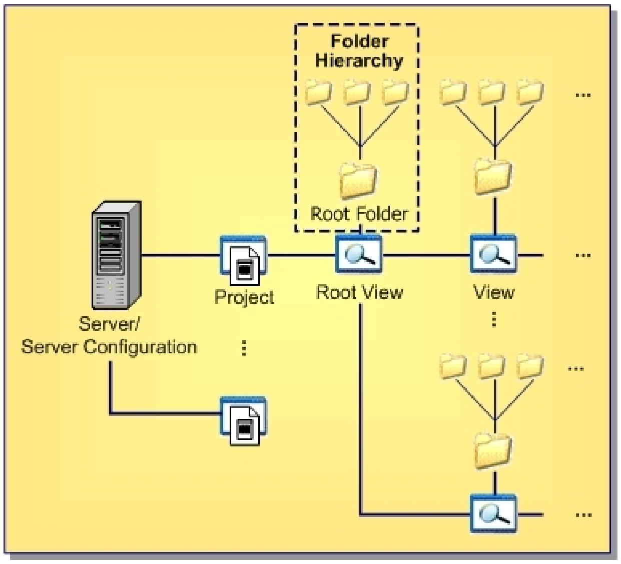
View-level Hierarchy

The root view and every child view has one application folder as a root folder.



Folder-level Hierarchy

An application root folder can have any hierarchy of child folders. This is called the folder hierarchy.



Guidelines for Keeping Projects Autonomous

The time-honored programming tenets of high cohesion and low coupling apply to StarTeam projects as well. The more independent your StarTeam projects are, the easier they are to secure, administer, and even separate from the original StarTeam configuration if necessary. Keeping projects autonomous means keeping cross-project links and shares to a minimum, avoiding them completely if possible.

Below are some guidelines for deciding what should be in the same project:

- A project should be used to manage the life-cycle artifacts of a cohesive application set or application component set. Examples are a commercial software product or a foundation library package. Multiple application or component sets can be managed in a single project if they are interrelated and generally enhanced and released together.

- A project should include all of the artifacts required to manage the life-cycle of the supported applications or components. This includes early life-cycle artifacts such as requirements documents, modeling diagrams, and design documents, as well as construction phase artifacts such as source files, images, and resource files; and later-phase artifacts such as test scripts and applications.
- A project should include all artifacts authored in life-cycle phases as well as non-authored artifacts required to perform authoring. This includes, for example, all files authored by the IDEs such as workspace/project files, source code, and resource files. It also includes "input files" such as `.h`, `.lib`, `.jar`, or `.dll` files that are authored or generated elsewhere but required by the project's IDEs or build processes. Input files may originate from third parties or from other projects in the same or other StarTeam configurations. (Transferring artifacts from one project to another is discussed further later).
- Files directly generated from authored files such as .obj, .class, and .lib files generally do not need to be checked into the project. However, it is common practice to check in "final" binaries such as `.jar`, `.war`, and `.exe` files that are delivered to other projects, engineering test, QA, or other deployment phases. The need to place generated files under version control is highly dependent on your own development, testing, and release methodologies.
- A project should have a long-term view of the products or components it supports. That is, it should house all artifacts generated over multiple iterations through the lifecycle. This means that the project supports multiple versions of its applications or components, representing the complete history of those modules.
- StarTeam works best when "work" artifacts (change requests, tasks, topics, and requirements) related to a project's files are stored in the same project. This allows, for example, a change request to be entered, tracked, and linked to the files in the same project to which the change request is related. This approach requires some special considerations for activities such as "change request triaging" and cross-project reporting. These issues are discussed later.

Some customers have attempted to use projects to separate development phases (for example, design and development) or development artifact types (like files and change requests). The artifacts are then interrelated by sometimes heavy use of links or shares. However, experience has found that copious use of shares – especially cross-project shares – results in difficulties in version control, reporting, security, and even performance. We suggest that artifacts related to the same applications and components, even though of different types/life-cycle relevance, should be managed in the same project.

Scenario 1: A Simple Client/Server Application

A commercial software application consists of a server written in C++ and a single client, also written in C++. Furthermore, the client and server modules share a fair amount of source code and IDE projects that generate common DLLs. The client and server modules are generally enhanced and released together.

In this scenario, a single StarTeam project should be used to manage the combined files of both the client and server modules. The sharing of source code and shared release schedules suggest that the modules are cohesive parts of a single application. Requirements, design documents, change requests, and other artifacts required for all life-cycle phases of the client and server modules should be managed in the same project.

Scenario 2: An Independent Client Module

A new Java client application is developed that uses the same server described in Example 1. Building and compiling the Java client requires some of the header files and one of the DLLs used by the server to produce a JNI wrapper, but no other source files. Furthermore, the Java application accesses other third-party servers and is generally enhanced and released on independent schedules from those used by the client/server modules.

In this scenario, it is reasonable to use a separate StarTeam project to manage the Java client's artifacts. The latest header files and generated DLL needed by the Java client are checked into a "external components" folder by the build process used in the client/server project. All change requests, tasks, and other life-cycle objects related to the Java client are managed in the same project.

Scenario 3: A Complex Financial Application Suite

A complex application suite consists of a set of foundation components and nearly 100 separate applications, divided into five functional areas: accounting, insurance, forecasting, etc. The applications are developed by different teams and all use the foundation components, which are jointly maintained by the development teams. Applications within a functional area are highly interrelated, but applications between functional areas are fairly independent. The foundation component library is enhanced and released on its own schedule, but the entire application suite is released as a commercial product in coordinated releases.

Although the entire application suite is interrelated, multiple projects should be used due to the overall application suite size. The foundation components are managed in one project, and each of the five functional areas utilize a separate project to manage the corresponding applications (six projects total). The foundation project is enhanced, built, and then “deployed” to each of the functional area projects by checking in generated jar files. Each development team generally opens only one project to perform their normal work. However, a special build script (using the StarTeam SDK) is used to extract files from multiple projects and generate “whole suite” builds. The build script also automates the management of common view labels and promotion states across projects.

Cross-Project Activity Support

Regardless of how you partition your projects, you may find that certain life-cycle activities span multiple projects. Some examples of these scenarios and how they can be addressed are provided below:

Multi-project Builds

Build procedures that require files from multiple projects can use the StarTeam SDK, which can open multiple projects at the same time. Alternatively, iterative calls to the StarTeam command line tool (stcmd) can be used to check out files from each of the required projects.

Defect Triaging

When a new defect is discovered, it is often entered as a change request before the module that must be repaired is known. This means that, if change requests are normally managed in the project containing the files that will be modified, a paradox exists in determining where to create the change request in the first place. A project leader or other person usually “triages” the change request by assigning it to whoever he or she thinks should handle it. As the change request is analyzed and addressed, the module that is ultimately modified to address it may not be known for awhile. A simple solution for this scenario is to place all change requests in a well known place (perhaps a “new defects” project) and “move” (not copy) them to the appropriate project as needed.

Cross-project Reporting

Currently, StarTeam does not provide built-in, cross-project reports. Consequently, if you want to generate reports such as “all open change requests across all projects” or “cross-server file metrics”, your best option is to use Datamart to harvest and report on change requests from multiple projects and even multiple configurations. Another option is to use the StarTeam SDK to write custom reporting applications.

How to Handle Cross-Project File Dependencies

When projects are highly cohesive, cross-project dependencies will be minimal, yet cross-project file relationships may still occur. Some files, either authored or generated, may need to be used in multiple projects.

The impulsive way to handle this situation may be to share the co-dependent files from one project to another. On the surface, this approach works, but experience has shown that cross-project sharing is problematic for several reasons:

- If a shared item’s behavior is configured to “floating”, changes from the parent project flow into the child project immediately, sometimes at inconvenient times. Many StarTeam users find that they need to manage the update propagation on a more scheduled basis.

- If a shared item's behavior is configured to a specific timestamp, it must be occasionally adjusted to allow updates to propagate. This makes the shared item read-only, and continually adjusting the configuration timestamp for a large number of shares can become burdensome.
- If a shared item's branch-on-change property is set to "true" (perhaps accidentally), and a change is made to the child share, the object will branch in the child project. This severs *automatic promotion* of changes from the parent item to the child share. If the child share is a folder, an innocuous change such as modifying the working folder will cause the folder object to branch
- When an update is made to an item, the entire share tree must be locked. As the share tree grows, update performance is impacted proportionately.
- Normally, when obsolete views and projects are deleted, the StarTeam purge utility can be used to return unused database records and archive files. However, shares used by a deleted project or view can prevent the purge utility from achieving the expected reduction in database and archive size. In short, it might not be possible to reduce the size of servers that use cross-product sharing.

Because of these reasons, other techniques have proven to be more effective at handling cross-project file dependencies. Below are some alternatives to sharing that work in different situations. In the end, shares still may be the most appropriate way to expose files from one project to another, but the approaches below should be considered first.

Deployment Approach

If a project "owns" a set of files that must be checked into other projects, you can establish a process in which the files are periodically "deployed". This means that the file set is checked into the target project(s) on an as-needed basis, perhaps by a script. Often, a build script is a good place to embed the deployment task, especially if the file(s) to be deployed are generated by the build. Keep in mind that checking in the same file multiple times does not (generally) increase the size of the vault, each unique file revision is only stored once.

Configuration Approach

Sometimes the co-dependent files don't need to be checked into each project, but they need to participate in a common process such as a build or delivery process. In these cases, a simple configuration file (for example, XML) can be constructed that defines the files that participate in the process. If the file is checked into an established location and updated when the configuration changes, then build, release, or other scripts can check out the configuration file, parse it, and base their processing on its instructions.

Link Approach

In lieu of shares, links can be used to connect objects across servers. Links do not possess many of the problems exhibited by shares, and they can be pinned, unpinned, and moved to refer to different object revisions. The downside of using links is that they are un-typed and possess no properties other than a link comment to identify their purpose.

Creating Projects

1. Start the StarTeam Cross-Platform Client.
2. Choose **Project** > **New** . The **New Project Wizard** opens.
3. Select a server configuration for the project from the **Server** list.

   Note: If the server you want is not in the **Server/Project Tree**, click **Add Server** and add it using the **Add Server** dialog box.
4. Click **Next** to continue.
5. Type the name and description for the new project in the **Project Name** and **Project Description** fields, and click **Next**.
6. Browse to the folder on your computer that will be the default working folder for the project root folder.

If the working folder does not exist, type the folder path and name and the **New Project Wizard** will create it.

Note: The default working folder must point to a location that is physically discrete for each user, such as a drive on your local computer or a personal directory on a shared file server.

7. Click **Next**.
8. Optionally, if the working folder has child folders, select any child folders you do not want added to the project and click **Exclude**.

   Tip: Click **Reset** to include the previously excluded folders.

9. Click **Finish** to complete and open the project.

   Note: After you create a project, the client window displays a hierarchical **Folder Tree** of folders in the current view of the project. You can add other folders, if desired.

Name (Project Properties Dialog Box)

Project > **Properties** > **Name**

Describes the options on the **Name** page of the **Project Properties** dialog box.

Name	Specifies the project name.
Description	Specifies the project description.
Created By	Displays the name of the person who created the project.
(Created) On	Displays the date on which the project was created.
Type	Displays the project type.

Options (Project Properties Dialog Box)

Project > **Properties** > **Options**

Describes the options on the **Options** page of the **Project Properties** dialog box.

Keyword Expansion	Enables keyword expansion.
Expand Keywords For These File Extensions	Restricts keyword expansion to files with the specified extensions.
Require Revision Comment When Files Are Checked In	Specifies that revision comments are required when checking in files to the project.
Require Exclusive Lock When Files Are Checked In	Specifies that all files checked in the must be exclusively locked to enable check-in. If developers are using an application integration for a development environment, such as StarTeam Visual Studio Plugin, they cannot check in files from that environment if both the **Require Exclusive Lock When Files are Checked In** check box in the **Project Properties** dialog box, and the **Use Non-exclusive Locks in Integrations** check box on the **Personal Options** dialog box (**File** tab) are selected. In this situation, uncheck **Use Non-exclusive Locks in Integrations** to check files in.
Mark Unlocked Working Files Read-Only	Specifies that all unlocked working files are marked as read-only if they are checked in, checked out, or unlocked when file locking is required. This option applies to files that are unlocked in the application or in application integrations with third-party applications.

The project property overrides the identical **Mark Unlocked Working Files Read-only** personal setting.

Default Types (Project Properties Dialog Box)

Project > **Properties** > **Default Types**

Describes the options on the **Default Types** page of the **Project Properties** dialog box.

Select the item types to be included by default when creating a new view in a project. The new view will include items of the selected types from the parent view.

You can select any or none of the following item types:

- Change Request
- File
- Requirement
- Task
- Topic

Process Rules (Project Properties Dialog Box)

Project > **Properties** > **Process Rules**

Describes the options on the **Process Rules** page of the**Project Properties** dialog box.

Require Selection Of Process Items When Files Are Added Or Checked In

Enforces selection of process items when files are added or checked in.

Permit Selection Of Change Requests As Process Items

Permits the selection of change requests as process items, and restricts them based on which status fields are checked in the **Restrict Status To** check boxes:

- If **Open** is checked, only change requests with a status of **Open** can be used as process items.
- If **In Progress** is checked, only change requests with a status of **In Progress** can be used as process items.
- If both **Open** and **In Progress** are checked, only change requests with a status or **Open** or **In Progress** can be used as process items.
- If neither**Open** and **In Progress** are checked, all change requests can be used as process items, regardless of their status.

Permit Selection Of Requirements As Process Items

Permits the selection of requirements as process items, and restricts them based on which status fields are checked in the **Restrict Status To** check boxes:

- If **Approved** is checked, only requirements with a status of **Open** can be used as process items.
- If **Approved** is unchecked, all requirements can be used as process items, regardless of their status.

Permit Selection Of Tasks As Process Items

Permits the selection of tasks as process items, and restricts them based on which status fields are checked in the **Restrict Status To** check boxes:

- If **Ready To Start** is checked, only tasks with a status of **Ready To Start** can be used as process items.
- If **In Progress** is checked, only tasks with a status of **In Progress** can be used as process items.

- If both **Ready To Start** and **In Progress** are checked, only tasks with a status or **Ready To Start** or **In Progress** can be used as process items.
- If neither**Ready To Start** and **In Progress** are checked, all tasks can be used as process items, regardless of their status.

Enable Enhanced Process Links

Enables the client to use enhanced process links.

If checked, the process item (that is, the item specified as the reason for making a given set of changes) is distinguished from the task that represents the act of making the associated changes in a particular view. Changes are linked to the process item indirectly, through a workspace (check-in) change package which is automatically created by the client.

If unchecked, standard links are used where the source of a process link is itself a process item. That is, if a given item is specified as the reason for a change, then process links are created directly from that process item to each changed file or folder.

Editors (Project Properties Dialog Box)

Project > **Properties** > **Editors**

The check box options on the **Editors** page of the **Project Properties** dialog box.

Use Alternate Property Editor For Files

Specifies that for files, StarTeam should use the specified alternate property editor in the corresponding field.

Use Alternate Property Editor For Change Requests

Specifies that for change requests, StarTeam should use the specified alternate property editor in the corresponding field.

Use Alternate Property Editor For Requirements

Specifies that for requirements, StarTeam should use the specified alternate property editor in the corresponding field.

Use Alternate Property Editor For Tasks

Specifies that for tasks, StarTeam should use the specified alternate property editor in the corresponding field.

Use Alternate Property Editor For Topics

Specifies that for topics, StarTeam should use the specified alternate property editor in the corresponding field.

Note: Your company must use StarTeam Enterprise Advantage to be able to use APEs. For more information creating APEs, refer to the *StarTeam Extensions User's Guide*.

Assigning Access Rights to Projects

Project-level settings define access rights for all views, child folders, and items in the project. A user will not be able to open a project if you deny any of the following to a user or to all the groups to which the user belongs:

- The ability to see the project.
- The ability to see the initial (or root) view of the project.
- The ability to see the root folder of the initial view of the project.

Note: It is critically important to define a complete set of project-level access rights for all groups with view rights to a project. By default, groups with view rights have complete access to categories that have no rights defined. Therefore, it is best to specify access rights across all project rights for all groups. Any time you create one grant record for a node, you should also create a grant record for that node for every group that will access the project at this level. It is also a good idea to include Administrators on each node. Then if permissions are ignored, an administrator can still change access rights, and so on.

1. Open the project and select **Project** > **Access Rights** . The **Project Access Rights** dialog box appears.
2. Select a node. It is best to start with the **Project** node and work down.
3. Click **Add** to add a user or group. The **Assign Access Rights To** dialog box opens.
4. Select a user or group. Users are listed by their user names and groups are listed by their position in the group hierarchy, except for the **All Users** group.
5. Select the **Grant** option button.

   Note: Never select the **Deny** option button unless you are creating an exception.

6. Click **OK**.
7. Select/clear the appropriate check boxes for the user or group.
8. Repeat steps 3-7 as required for additional nodes

Granting Project-Level Access Rights

This section provides information about setting access rights at the project level. It illustrates this information by using an example with three user-defined groups: **Developers**, **Testers**, and **Others**. (These groups are in addition to the **All Users**, **Administrators**, **System Managers**, and **Security Administrators** groups that come with the StarTeam Server.) The example also assumes that the **All Users** group is larger that the **Others** group.

Project Node

Assume that you decide that only members of the **Administrators** group should control the project and create a grant record. This record prevents anyone who is not a member of the **Administrators** group from seeing the project, unless privileges apply. As a result, no one else can access and work with the objects in this project.

Note: Although members of the **Administrators** group require all access rights for the project, you may decide to omit them from the Users and Groups list if they have group privileges. Normally, this is acceptable. However, if the server configuration is set to ignore privileges, you must specifically grant the **Administrators** group all project access rights.

Next, you must assign the correct rights to the every other group that needs to access this project. Because keyword expansion is a project property, the **Developers** group needs to have the rights to see the project and modify its properties. However, they probably do not need to delete the project or change its access rights. The **Testers** and **Others** groups need to see the project and its properties, so you should select only the **See Object And Its Properties** check box for these groups.

View Node

View access rights at the project level apply to all views that now exist or will be created for this project in the future. Members of the **Administrators** group need all view rights. They may be assigned these rights or receive them because of their privileges. The **Developers** and **Testers** groups need to see and modify view properties and perform operations on labels. They do not need to create or delete views, manage promotion states, or change view access rights. The **Others** group needs to see the view, but requires no other rights.

Promotion State Node

The Promotion State node is not important in this example.

Child Folders Node

For access rights to child folders at the project level, the **Administrators** group may need all rights. They may be assigned these rights or receive them because of their privileges. The **Developers** and **Testers** groups probably do not need to delete folders, share or move folders, change folder behaviors or configurations, or change folder access rights. You may want the **Others** group to only see the folders, their properties, and their histories.

Item Nodes

We do not recommend creating only one grant or deny record for a given node. The following section illustrates how project-level item access rights work, using files as an example.

If only the developers need to access files, you can grant only the **Developers** group all file access rights at the project level.

With this setting, only members of the **Developers** group have access rights to any files, regardless of the view, folder, or file. As a result, only developers can see or perform operations on any files. Members of the **Testers** and **Others** groups see only the files that they have in working folders, but the status of these files appears as **Not In View**. However, by default, one exception exists by using **Privileges**. If groups other than the **Developers** have one or more privileges that allow them to see, modify, define, or perform other actions on a file, members of those groups have access to the files regardless of the access rights settings. For example, the default settings for the **Administrators** group grant all privileges to this group. Therefore, members of this group can perform any file operations.

You can stop the server configuration from checking for privileges.

If you give only the **Testers** and **Developers** groups access to other types of items (change requests, requirements, tasks, and topics), the same exceptions apply. However, other groups will want to see these types of items, so you will need to grant these groups some access rights.

Project Access Rights

The following table describes the generic object rights for a project. To display the **Project Access Rights** dialog, select the **Project** > **Access Rights** command. The right to create a project is set as a server access right.

Generic object rights

See object and its properties

See this project and view its properties by selecting **Project** > **Properties**.

Modify properties

Change the properties for this project. The project properties that can be modified are name, description, keyword expansions settings, alternate property editor (APE) settings, process rules settings, requiring unlocked files to be read-only, and several settings that affect users (for example, requiring revision comments to be entered when a file is checked in).

Delete object

Delete this project from its server configuration.

Change object access rights

Change the access rights for this project. If you change this setting, be sure that you remain one of the users who can change access rights.

Opening Existing Projects

Before you can access an existing project view, you must have access to the appropriate server, be logged on, and open the project.

1. Do one of the following:
   - Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
   - Select **Project** > **Open** . The **Open Project** wizard appears.
2. Click the plus sign in front of the server name or double-click the name of the server configuration on which the project is located.

- If you have already logged on to this server configuration, skip to the next step.
- If you are not logged on, the **Log On to [server configuration] [project name]** dialog opens.

3. Type your **User Name** and **Password**. You must have the necessary access rights to continue. Passwords are case-sensitive and may have length restrictions. See your StarTeam administrator for details.
4. Once you are logged in, the **Open Project** wizard displays a list of projects for the selected server configuration. Do one of the following:
   - Select a project name, then click **Finish** to open your project.
   - Double-click the project name to select a specific view of that project.
   - Select the project name, then click **Next** to select a specific view of that project.
5. If the **Select View** dialog opens from the **Open Project** wizard, select a name from the **View** list and click **Finish** or simply double-click the view name to open your project in that view.

Note: If the **View** icon is disabled, you do not have access to that view.

Opening Projects with Shortcuts

If you will be accessing a specific StarTeam project view frequently, you may want to create a shortcut for it on your desktop. Double-clicking the shortcut starts StarTeam and opens the view associated with the shortcut.

1. Click **Project** > **Open Shortcut**. The **Open** dialog box appears.
2. Select the shortcut name and click **Open**.

Note: The view configuration is also saved as part of the shortcut.

Changing Project Names or Descriptions

If you have the appropriate access rights, you can use the **Project Properties** dialog box to review or change the project name and description.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Name** tab.
3. Type a new project name in the **Name** field.
4. Type a new project description in the **Description** field.
5. Click **OK**.

Saving Projects as Shortcuts

If you will be accessing a specific StarTeam project view frequently, you may want to create a shortcut for it on your desktop. Double-clicking the shortcut starts StarTeam and opens the view associated with the shortcut.

1. Log on to StarTeam and open the project view window for which you wish to create the shortcut.
2. Select **Project Save Shortcut As** or click **Save Shortcut** on the toolbar. The **Save As** dialog box opens.
3. Use the default name or type another name for the shortcut in the **File Name** field. Be sure to keep the `.stx` extension.
4. Select a location, usually your desktop, for storing the shortcut.
5. Click **Save**.

Viewing or Modifying Project Properties

Each project can be configured with properties that allow you to specify the project name, enable keyword expansion, require revision comments on check-in, specify file locking behavior on check-in, define process rules and process items, and specify alternate property editors.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Select a tab containing properties you want to view or change and make the changes.
3. Click **OK**.

Deleting Projects

To delete a project, you must have the delete privilege or access right. Be absolutely certain that you want to delete a project and its folders because you and other members of your team will no longer be able to access any item in the project.

If other users are connected to the project at the time it is deleted, they will receive a message the next time they initiate a project or view command.

Deleting a project does not remove any data from the StarTeam Server database. However, items that are not shared will no longer be accessible.

1. Select a folder from the folder tree.
2. Select **Project** > **Delete** .
3. A message displays asking you to confirm the deletion. Click **OK**..
4. A dialog box opens requesting you to enter the exact name of the project that you wish to delete. Type the name in the field provided.
5. Click **OK**.

Setting Active Process Items

Setting an active process item is a convenient way of saving time when you know that you will be adding files or checking them in later. When you have an item selected on the upper pane, making it the active process item is a simple operation.

You can make a selected change request, task, or requirement the active process item for the current view, an open view on the same server, or a different view on the same server.

Note: You can only specify one active process item for each view. Setting a second active process item for the same view at one time clears the first one.

1. Select a folder from the folder tree.
2. Click the **Change Request**, **Requirement**, or **Task** tab and select the item you want to use as the active process item.
3. Do one of the following:
   - Right-click and choose **Set Active Process Item** > **Current View** to choose the current view.
   - Right-click and choose **Set Active Process Item** > **[view name]** to choose from the listed opened views on the server.
   - Right-click and choose **Set Active Process Item** > **Select View** to open a dialog box and choose any other view on the server.

   Note: You can also set the currently-selected item as a process item by using the **Use As Active Process Item** button on the toolbar.

The active process item you selected is used by default when you add files or check them in. However, you can override this default and select another appropriate item when adding or checking in files.

Tip: After you finish with a process item, you should right-click it and choose **Clear Active Process Item** so that it cannot be accidentally reused. That removes the information from the status bar and keeps the process item from reappearing in the **File Add** or **File Checkin** dialog boxes.

Establishing Process Rules for Projects

Establishing a system of process rules allows you to:

- Require that process items are used every time files are added or checked into the project.
- Stipulate that only certain types of items with specific statuses can be used as process items in the project.
- Enable the use of enhanced process links for the project.

Note: To set process rules, you must have the access rights required to change project properties. Usually, only team leaders and administrators have these rights. You must also verify that project users have the rights to see and modify items in the project view, to create and modify links on files and process items, and to create tasks and link to tasks if using the enhanced model.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Process Rules** tab.
3. Select the **Require Selection Of Process Items When Files Are Added Or Checked In** check box.
4. Select the type you want to allow for use as process items.
5. You can define the use of the type as a process item in the **Process Item Details for <Type>** section.

To permit the use of any type as a valid process item:

1. Select the desired **Type**.
2. Specify the **Active States** that are permitted to be used as a process item during commit.
3. Specify the **Closed State** that will be used to mark the process item as completed upon successful check-in.
4. Add the <Type> as a valid process item type.

Note: Some StarTeam integrations do not recognize process rules and will ignore them.

Configuring Projects to Use Alternate Property Editors

If your company has licensed StarTeam Enterprise Advantage, you can use alternate property editors (APEs). APEs are Java forms created specifically for your company to support a corporate process. Workflow processes are created for use with the forms. APEs can be created for the File, Change Request, Requirement, Task, and Topic components.

APEs use the StarTeam SDK to access a StarTeam server configuration. APEs can be customized because they are implemented in standard programming languages. Sample property editors for several StarTeam components are included with StarTeam Enterprise Advantage.

By default, every project uses the standard property dialogs. When an APE is ready for use, you must change the project properties so that all the views in a project will use the APEs. Once set, every view in the project must use an APE instead of the standard dialog. However, each view can use a different APE that you have checked into the appropriate view-specific subfolder in the Projects folder of StarFlow Extensions project.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Editors** tab.
3. In **Type**, select the component for which you want to create an APE.
4. Click **Browse** to navigate and find the correct APE. Generally, you enter the word `Locator` followed by the name of the APE. `Locator` is a program on each client workstation that distributes code and XML updates to client workstations.

When `Locator` is specified and StarTeam requests the APE, `Locator` looks for the StarTeam Extensions project managed by the server configuration. If the project exists and contains an APE with the specified name, Locator copies the tip revision of that APE and its related files to the client workstation (if they are not already there). Then the APE will be used instead of the StarTeam standard dialog for that type of item.

Caution: Take care when setting the APE for a project in production because the new setting takes effect immediately. It is important to test any changes in an APE prior to making this editor available to a wide audience.

5. Click **Add**, **Update**, or **Delete** depending on which action you want to perform to configure the project.
6. Click **OK**.

Displaying Location References

Because of manual sharing and because views are children of other views, a folder or item can be associated with more than one project, view, or parent folder (within the same server configuration). Each instance of the folder or item has a reference to its tip revision.

To view folder references, you open a separate dialog; to view item references, you use the project view window and the **Reference** tab on the lower pane.

You can view references for any of the following:

- Folders and past revisions of a folder.
- Items and past revisions of an item.

Enabling Keyword Expansion

By enabling keyword expansion for a project, you can embed keywords within text files. These keywords are automatically expanded during file check-outs, to provide file and revision information within the file. You should use only one keyword per line.

Keyword expansion and EOL conversion work correctly for UTF-8 unicode files, which previously could be corrupted on check-in. However, EOL conversion is not supported for UTF-16 or UTF-32 unicode files. The Cross-Platform Client allows you to select a file encoding as you check out a file.

StarTeam supports Log and History Keywords from the StarTeam Server and from the MPX Cache Agent.

Caution: Never use a keyword in a revision comment, as it will be expanded during the keyword expansion process.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Options** tab.
3. Check the **Keyword Expansion** check box to enable keyword expansion and use keywords in your text files.

   This check box applies to files added or checked in from StarTeam or from StarTeam integrations with third-party applications.
4. In **Expand Keywords for These File Extensions** field, type the file extensions (for example, `.bat`, `.cpp`) for which you want to use keywords.

   You can use a space, comma, or semicolon as keyword delimiters. The file extensions list can contain a maximum of 254 characters. If you leave this text box blank, no keywords will be expanded.

StarTeam Keywords

By enabling keyword expansion for a project, you can embed keywords within text files. These keywords are automatically expanded during file check-outs, to provide file and revision information within the file. You should use only one keyword per line.

Item **Description**

`$Author$`

User who checked in the revision.

`$Date$` and `$DateUTC$`

Date and time stamps for the revision. `$DateUTC$` is the same as `$Date$` except that a UTC time replaces the local time. UTC times end in a “Z,” which makes them readily identifiable.

`$Header$` and `$HeaderUTC$`

Combinations of Workfile, Revision, Date, and Author. `$HeaderUTC$` is the same as `$Header$` except that a UTC time replaces the local time. UTC times end in a “Z” which makes them readily identifiable.

For Java users, `$Header$` can cause problems if `\u` (for unicode) appears in the expanded header. For example, suppose that a file named `foo.java` is stored in `D:\util`. The first time you compile it with $Header$, the header is expanded to:

```
$Header: D:\util\foo.java, 1, 7/27/11 11:05:48 AM, StarTeam
Server Administrator$
```

Even though this header is contained in a Java comment, the Java compiler always looks for `\u`. The second time you compile `foo.java`, a compiler error occurs.

`$History$`

Added to the source file, typically within a comment field. The `$History$` keyword creates a history record for the latest (tip) revision and places the information after the keyword. For example, after the file is checked-out for the first time, the file would contain the following:

```
// ...
// $History
// 1 YourProject 1.0 2011-11-19 00:06:57Z Joe Smith
// This is a revision comment.
// $
```

StarTeam adds the history information to the file and places it outside of the “$” signs so that it becomes a versioned part of the file. Since the history records become part of the versioned file content, you can delete extra or excessive history records at any time. If this file is modified and checked-in, any subsequent check-out adds one additional record immediately after the `$History$` keyword:

```
// $History
// 2 YourProject 1.0 2012-06-01 00:06:57Z Joe Smith
// This is a another revision comment.
// $
// 1 YourProject 1.0 2011-11-19 00:06:57Z Joe Smith
// This is a revision comment.
```

StarTeam uses the MD5 and local file timestamp of the post-expanded file for sync records. Accordingly, when you check-out a tip revision, StarTeam reports a *current* status for the file even though a compare tool would show an additional history record in the locally-stored file as compared to the version of the file saved on the StarTeam Server.

Important: Do not delete the standalone `// $` line. StarTeam places the most recent historical revision information within the `// $` delimiter. The rest of the revision entries are outside of this delimiter. StarTeam does not recognize the `$History$` keyword if you remove this line.

`$Id$`

Similar to `$HeaderUTC$` except that it is a combination of Workfile, the branch revision number (preceded by "v" for version; for example, `v 1.2.1.0`), UTC time, and Author. The branch revision number is in dot notation.

Item	Description
`$Locker$`	User who has the file exclusively locked (if any).
`$Log$`, `$Log[x]$`, `$LogUTC$`, and `$LogUTC[x]$`	File change history. `$Log$` is a special keyword because it expands to a multiline entry. The `$Log$` keyword expands to include information for each revision of the file. Revision history includes Revision Number, Date, Author, and Reason for Check In. Use `$Log$` to retain entries for each revision within the file. Use `$Log[x]$` to limit the number of revisions for which entries are retained. Replace `x` with the number of entries to be retained. For example, `$Log[8]` saves the entries for the most recent 8 revisions. If you replace `x` with a number less than 1 or with a non-numeric character, StarTeam ignores `x` and retains all entries (as with `$Log$`). `$LogUTC$` and `$LogUTC[x]$` are the same as `$Log$` and `$Log[x]$` except that a UTC time replaces the local time. UTC times end in a “Z”.
`$NoKeywords$`	Turn off keyword expansion for the rest of the file.
`$Project$`	Name of the project.
`$Revision$`	Revision number (an integer).
`$Folder$`	Name of the folder.
`$Workfile$`	Unqualified name of the working file (for example, `foo.cpp`).

⚠ **Caution:** Never use a keyword in a revision comment, as it will be expanded during the keyword expansion process.

Requiring Exclusive Locks for Check-ins

With this application, you cannot force users to lock files before they make changes. However, if you have the required privileges, you can require all users to conform with the policies and processes of your company by exclusively locking files before they check them back into the application.

Although this requirement helps users to avoid merge situations, they still must:

- Notice whether a file is already exclusively locked by another user before they check it out to work on it.
- Lock each file before making changes, to alert other users to their intentions.
- Be sure that the status for each working file is `Current` to avoid changing older revisions of the files. If a file status is not `Current`, the file must be checked out before any changes are made.

To force users to lock files before checking them in

1. Choose **Project** > **Properties** .

   The **Project Properties** dialog box opens.
2. Click the **Options** tab.
3. Check **Require Exclusive Lock When Files are Checked In**.

   When this option is selected, only a person who has exclusively locked a file can check it in.

   Note: If developers are using an application integration for a development environment, such as StarTeam Visual Studio Plugin, they cannot check in files from that environment if both the **Require Exclusive Lock When Files are Checked In** check box in the **Project Properties** dialog box, and the **Use Non-exclusive Locks in Integrations** check box on the **Personal Options**

dialog box (**File** tab) are selected. In this situation, uncheck **Use Non-exclusive Locks in Integrations** to check files in.

Requiring Revision Comments

Administrators can force users to supply a check-in reason, however, by adjusting the properties for the project. This requirement will apply no matter which way users perform the check-in.

1. Choose **Project** > **Properties** .

   The **Project Properties** dialog box opens.
2. Click the **Options** tab.
3. Check **Require Revision Comment When Files are Checked In**.

   Note: This check box applies only to the application, not to integrations.
4. Click **OK**.

   From this time on, the **Check-in** dialog box will always open when users check in files, and they will need to type text in the **Reason for Check-in** field before completing the operation.

Viewing Connection Properties

Reviewing the connection properties lets you verify the server, address, end point (port), logged on user, and logon time.

1. Select **Project** > **Connection Properties** . The **Connection Properties** dialog box appears displaying all the server connection property values.
2. Click **OK**.

View Configuration and Management

A view is a window through which you can access a subset of the artifacts in a project. Furthermore, a view uses items to reference artifacts and manage access to them. Depending on how it is configured, a view can:

- Serve as a sub-project for a specific development or maintenance activity.
- Be a read-only or updateable subset of another view.
- Be used for other purposes.

The good news is that views have tremendous flexibility to serve many needs. The bad news is that you can get unexpected or undesirable results if you don’t understand how the different view types work.

This section describes each view type, how it behaves, and when you might want to use it. It also discusses the roles views can fulfill for specific development activities and suggests practices for managing changes within a view and for propagating changes from one view to another.

Overview of Views

When you create a new project, the server creates an initial or root view of that project with only the "File" type pre-selected as a default for new views. Users can still change the project properties after the project is created, and they can change the item types included for any given new view. However, if the user changes nothing, by default new views will only include files when they are created.

This initial view, which has the same name as the project, consists of the root folder, to which you will add child folders, files, and eventually, more. It is always read/write. The root view is ideal for collaborative development. because it is dynamic, showing all items in the project as they change.

To accommodate both user and project needs, however, the application enables you to create additional views of a project based on this view. These additional views, called child views, may contain some or all of the contents of the original view and may behave differently. For example, you might use child views to:

- Implement the same folder hierarchy for multiple releases of a product. If the root view is for current development, you maintain the folder hierarchy there and create new child views for each release at about the time that it ships. Maintenance would be done in the child view.
- Limit the portion of the project that certain team members see. Developers might need to see only the project's source code folder and its child folders; marketing personnel might need to see only the marketing folder and its child folders; and so on. Each of these views can have a different folder as its root.
- Support branching and parallel development. By branching files and other data in a new view, your organization can start to work on the 2.0 version of a product without hampering the creation of service packs for the 1.0 version.

Views represent configurations of items and support different development baselines of the same code base. It is common practice to promote changes from one release's maintenance view to the root view where the next release is being created. Views can be reconfigured to show items as they existed at an earlier point in time, or based on a view label or associated promotion state. Rollback views are read-only, as they show a precise state of the items and no longer permit changes.

Understanding Branching Views

A branching view is a view that permits branching—that is, the folders and other items in the view can branch or separate from the corresponding items in the parent.

Branching views serve many purposes; you can create a branching view to meet different needs from those of your main line of development. For example, you might create a branching view for a maintenance release or a custom version of your product. A branching view can also be used to keep an area of your project private until it is completed and tested. Then you can merge your changes into the main line of development when and where necessary.

A branching view should use a different working folder than that of its parent view. Using the same working folder for both views is not only confusing but can create status problems.

Item Branching Behavior

Given the appropriate settings, folders, files, and change requests in a child view can be branched, that is, can be separated from the corresponding item in the parent view. Folders and change requests branch when their properties change, while files branch when either their contents or their properties change (Requirements, tasks, and topics can never branch).

For each item, branching occurs a maximum of one time per view. For example, if a new item is added to the root view, its first revision has the dot notation `1.0`. Subsequent revisions become `1.1`, `1.2`, and `1.3`. Suppose this item is included in two child views created from the root view and that both of the child views are branching views. In one child view, if a new revision is made to the item, the item branches. This separation from its "parent" item in the root view is indicated by an addition of two numbers to the dot notation. If the parent item was `1.3`, the child item becomes `1.3.1.0`. The child item's next revision becomes `1.3.1.1`.

Now, suppose the corresponding item in the second child view is changed. Its dot notation must also change. Because `1.3.1.0` already exists, the separation of this item from its parent item gets the dot notation `1.3.2.0`. This child item's next revision becomes `1.3.2.1`.

The original item in the root view has the history: `1.0>1.1>1.2>1.3`. The next revision will be `1.4`. The item in one child view has the history: `1.0>1.1>1.2>1.3>1.3.1.0>1.3.1.1`. The next revision will be `1.3.1.2`. The item in the other child view has the history: `1.0>1.1>1.2>1.3>1.3.2.0>1.3.2.1`. The next revision will be `1.3.2.2`.

Whether or not a folder, file, or change request has the ability to branch depends upon its behavior settings.

- If the **Branch on Change** check box for an item is enabled and selected, the item can branch.
- If the **Branch on Change** check box for an item is enabled but not selected, the item cannot branch, but its behavior can be changed.
- If the **Branch on Change** check box for an item is disabled, the item has either already branched or its location is where it was added to StarTeam.
- If the **Branch on Change** check box for a folder is enabled but not selected, new items cannot be added. If you attempt to do so, an error message will appear, stating that the folder is read-only.

Note: Changing a folder's branching behavior does not affect the behavior of items in the folder. Items in a folder have their own branching behavior.

Items in Both the Parent and the Branching View

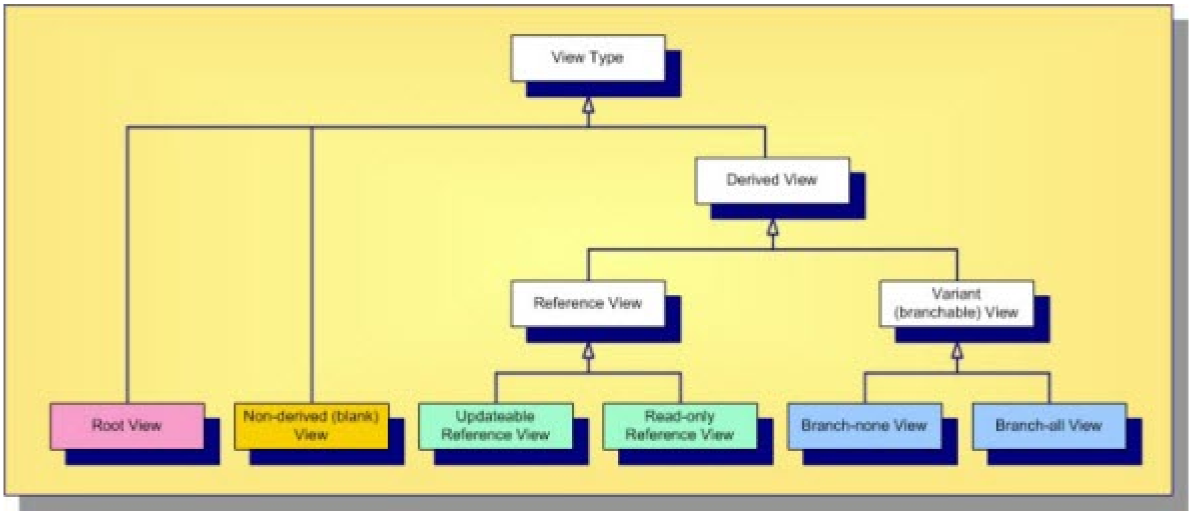
Depending on the folders included in the new branching view, certain items from the parent view appear in the new branching view. However, these "inherited" items will have no labels, as neither view labels nor revision labels move from view to view.

As a result, the workflow for change requests is affected in the following ways:

- If the **Last Build Tested** and the **Addressed In Build** fields in a change request have no values when the change request branches, their workflow is specific to the new view only.
- If the **Last Build Tested** and the **Addressed In Build** fields in a change request have build labels as their values (i.e., these fields are not empty or do not contain the value `Next Build`) when the change request branches, the branched change request retains those values. In the new view, these values can be changed, but only to the names of build labels that exist in the new view.
- If the **Addressed In Build** field contains the value `Next Build` when the change request branches, the `Next Build` value is replaced by the name of the next build label created in the parent view, not the next build label created in the new view.

View Types

As shown, there are six concrete view types. Aside from the special *root* view type and one *non-derived* view type, the rest are *derived* view types, which subdivide into *reference* and *variant* types, each of which have two concrete types.



The abstract and concrete view types are described below.

Root View A root view is the "main" view automatically created for each project. Initially, it receives the same name as the project, but you can change it. (It is often renamed "Main" to emphasize its role.) When first created, the root view has only a root folder whose name matches the view. It is the only view type that has no *parent*, it forms the top of the project’s view hierarchy.

Non-derived View This is also called a *blank* view. Like the main view, it initially has no items except for a root folder. Although a non-derived view has a parent view, it is not *derived* from that parent, which means it does not inherit the parent view’s items. You can add new items to it, or you can share items from other views into it. A non-derived view can be used for non-lifecycle activities, acting like a “scratch pad” that you build up one item at a time.

Derived View A derived view begins life as a subset or an exact copy of its parent view. One of the parent view’s folders (often the root folder) is chosen as the root folder of the derived view; hence the derived view starts as a window into the artifacts at which it is rooted. What you can do with the artifacts in a derived view depend on whether it is a variant or reference view.

Reference View A reference view is a derived view that is a pure subset of its parent view. A reference view does not have its own items; it uses the same items as its parent view. Consequently, updates made to a reference view (if allowed) are applied to the same items used by the parent view. This also means that reference views never have their own artifact branches; hence reference views are also called *non-branching views*. Reference views also do not have their own view labels, they share the same labels as their parents. Reference views are considered lightweight since they do not have their own items. A reference view can be updateable or read-only.

> **Updateable Reference View** An updateable reference view is a pure subset of its parent view with no added restrictions. If an item is updateable in the parent view, it is updateable in the reference view as well. Since the two views share the same items, changes in either view are immediately visible in both views. An updateable reference view is useful for exposing a portion of another view for security purposes. Instead of adding folder- and/or item-level security rights to the parent view, adding view-level security rights to an updateable reference child view is often easier to manage.
>
> **Read-only Reference View** A read-only reference view is a subset of items from the parent view that cannot be modified through the child view. A child view can *float* to the tip configuration of the parent view. In this case, it immediately reflects any changes made in the parent view to items it can see. Alternatively, a read-only reference view can be *pinned* to a specific configuration of the parent view: a timestamp, view label, or promotion state. A pinned read-only reference view reflects the state of the parent view (subset) as of that configuration. Once the child view is created, you can’t change the configuration on which it is based. However, in the case of labels and promotion states, the child view will follow changes made to those objects. For example, if new items or different revisions are attached to the label or promotion state in the parent view, the child view will immediately reflect that change. Floating and pinned read-only reference views are useful when you want to subset a view and ensure read-only access, for example with applications such as build scripts.

Variant View Unlike reference views, a variant view is not a pure subset of its parent view. Although a variant view may initially be created as an exact copy or subset of its parent view, it has its own items. In fact, when a variant view is first created, the parent view’s items (or subset thereof) are shared (copied) to the child view, initially referencing the same artifacts. When new items are added to the child view, they may not be automatically added to the parent view depending on the containing folder’s configuration. Furthermore, since it has its own items, the variant view’s items may be independently configured, which means they could branch. Consequently, variant views are also called *branching views*. Whether or not a variant view’s items are initially marked branch-on-change (BOC) is the major difference between the two variant view subtypes.

Branch-none View If a variant view’s items are initially configured with BOC set to “false”, it is referred to as a branch-none view. It acts a bit like a reference view, except that it doesn’t share labels with its parent. If the child view’s items are created with a pinned configuration, they will be read-only. If the item configurations float, updates through those items will float to the parent. This is why a branch-none view is sometimes called a “floating view”. However, because the child view has its own items, “moves” in the child will not propagate to the parent view. Because of the discrepancy between the propagation behavior of moves versus other updates, branch-none views can be very confusing. It is possible to create branch-none views and hand-tweak item configuration and branch-on-change, but this approach has proven tricky at best and sometimes disastrous. Consequently, we recommend against using branch-none views.

Branch-all View Branch-all variant views are the most commonly-used views. A branch-all view begins life as a copy of its parent view (or subset thereof), with branch-on-change set to “true” for all items (that point to branchable artifact types). A branch-all view’s item can be configured to float, causing changes in the parent view to float to the child view. However, branch-all views are far more useful when all items are configured to a specific timestamp or view label.

Of the six StarTeam view types, you will use main views and branch-all views with configured items (not floating items) the most. As detailed in the next section, each project will have only one main view, but you will use branch-all views to support both new development and maintenance activities. Reference views are occasionally used as a way to expose a read-only or updateable subset of its parent while simplifying security management. Blank views are rarely used, and branch-none views are not recommended.

View Roles

Technically, you could manage a project’s artifacts using only the root view. For example, you could:

- Add all new folders, files, change requests, and other items directly in the root view.
- Acquire locks as you modify items to prevent conflicts between users.
- Employ revision labels and process items to identify revisions related to the same logical change.
- Create view labels to mark milestones such as specific builds and releases.
- Use promotion states to propagate snapshots represented by view labels through a coordinated test and release process.

When your project is first getting started, or if you have a very small (and safe) change to make, direct modification to the main view is fine. Otherwise, direct modification increases the risk of breaking the build if you haven't staged your changes elsewhere first. The moment your team needs to work on two versions of the same file, module, or application at the same time, it needs a place other than the root view to do its work. This is the purpose of *development streams*. You need containers that support parallel development or maintenance activities that require different artifact branches within the project. The child views you create under the root view provide these containers.

Because version control and SCM products have existed for several decades, it should be no surprise that many different *patterns* have been developed for managing software development artifacts. These patterns affect the number of development streams you deploy and how you propagate changes between them. With StarTeam, this translates to the number and type of views you use and how they are organized.

Most of the child views in your view hierarchy should be branch-all views. To determine when you need a new view, where it should live in the hierarchy, and whether a different type of view would be more appropriate, you should consider the role that each view will fulfill. Based on our experience, the best way to use views is to consider the roles described in the next sections.

Main View: Home Base for Artifacts

The *main view*, also known as the *root view*, should contain the latest, approved revisions for the whole project. By latest, we mean that the main view should match your latest changes, ready for the next release. By approved, we mean that it should contain revisions that have undergone whatever verification checks your process requires: a complete build, unit testing, integration testing, etc. By whole project, we mean that it should not be a subset of the project's modules. In short, your main view should contain the latest, complete, production work, ready to be seen and used by everyone.

Implicit in this recommendation is that the main view should always be clean. That means it should be buildable and able to pass most if not all tests. Experience has shown that you will avoid many headaches by keeping your main view clean. To do this, all except for the simplest changes should be made in other views where they can be tested and fine-tuned until they're ready to be propagated to the main view. Once a new or modified revision has passed the point of no return—you're sure you won't change your mind—only then should it appear in the main view.

There is a subtle but important interaction between the main views and "share trees" you should be aware of. In general, the main view should contain the main (1.n) branch of each artifact. This happens automatically when you add a new artifact in the main view. StarTeam first creates a new 1.0 artifact and then connects it to the view and parent folder with a new item. But suppose you create a new artifact in a child view and then share the item "up" to the main view. Since the child view item was created first, it will be the "root share", and the main view item will receive a "child share" item (initially pointing to the same artifact). In other words, the item share tree will point in the opposite direction as the views. You probably won't notice a problem until you modify the item in the main view, causing it to branch, thereby pointing to a non-main branch (e.g., 1.n.1.m). In future attempts to propagate changes between the two views, you'll find that it gets harder to propagate changes correctly because the share tree is backwards. For example, "rebase" operations won't work correctly. What can you do about this? The solution is to use the view compare/merge (VCM) facility to promote new items to the main view. VCM understands the share tree issue and propagates new items using an operation called *reverse share*.

Note: Savvy StarTeam pre-2006 users are aware of this subtle "share tree direction" issue and employ custom solutions. Some customers simply add new items to the root view first and then share them down to child views. Other customers propagate new child view items to a parent view by first moving them up and then sharing them back down, this is a basic version of what a VCM reverse share does.

Activity View: Isolated Team Work Area

Suppose your team is charted with developing a significant new enhancement (or a set of enhancements to be developed together). By significant, we mean changes that will affect more than a few artifacts, take more than a few days, and/or involve more than a few people. You wouldn't want to make changes directly to the main view since:

1. You want to check in your changes often to ensure they are backed-up, progress can be tracked, etc.
2. It may be a while before your changes are sufficiently stable to begin testing. The way to isolate your team's work is with an *activity view*.

An activity view is a branch-all view created from a well-identified, stable configuration of the main view. If the enhancement work only requires a portion of the main view's modules, you might choose to root the activity view from something other than the main view's root folder. Typically, an activity view is created from a view label or promotion state, which causes it to contain items pinned to the same revisions in the main view as of that snapshot. As changes are made to the activity view, the corresponding items will branch and hence will not be visible to the main view. New items are also added to the activity view and hence not visible to the main view. As work progresses, the state of tip revisions in the activity view may not always represent a buildable release. But eventually, your team will build, test, fix, and finish its work, whereupon it is "promoted" to the main view.

Activity views typically have a limited lifespan: when the enhancement work is done and promoted, they can be deleted, usually after a certain period of time. Strictly speaking, a single activity view could be used for multiple enhancement activities, but there is an advantage to using separate views for each activity: if for some reason the activity must be cancelled, the activity view can be abandoned and eventually deleted. This isn't practical if the view contains work from multiple activities. Either way, activity views periodically require “rebasing”.

A variation of the activity view is an *integration view*, which supports integration activities for a large software project.

Release View: For Post-Release Maintenance Work

When you release a snapshot of your software, you'll probably have to maintain a development stream just for hot fixes, patches, service packs, and so forth. A view that fills this role is called a *release view* (or a *maintenance view*).

Like an activity view, a release view is a branch-all view created from the main view as of a specific snapshot. A release view is created after one or more enhancement activities have been completed and promoted to the main view. It represents a milestone where your software has been (or is about to be) released externally. The release view is almost always rooted at the main view's root folder, and it is usually created from a frozen view label to clearly establish the software configuration that was actually released. In fact, many organizations create the release view first and then build and deliver the software from the release view.

Since it is a variant view, a release view can receive changes: for example, to fix bugs. If these bugs must be propagated to the main view, they are promoted as in activity views. However, release views are generally not rebased to receive changes from the main view except for bugs fixed in the main view first that must be applied to the release view as well.

If you need to make significant changes to a release view (perhaps a service pack), it is acceptable (even advisable) to create an activity view as a child of the release view. You would then make the changes in the activity view, perform appropriate validation tasks, and then promote it to the release view. You might even want to use a sandbox view, which is discussed below.

Build View: Read-Only Windows for Build Scripts

Many StarTeam customers create build applications using simple build tools (for example: make, nmake), a commercial build product, or open source components (for example: Ant, CruiseControl). It is easy to integrate these tools with StarTeam due to the availability of its full-featured SDK and the availability of pre-built components such as StarTeam Ant tasks and a CruiseControl “bootstrapper” plugin for StarTeam. Build applications typically open a view, specify a snapshot configuration (timestamp, view label, or promotion state), and then check out the files they need.

In some cases, however, an organization may want to restrict the access of the build tool (or user) in comparison to the permissions other users have to the view. For example, you may want to guarantee that (a) the build tool can see only artifacts required by the build (and not design documents or other artifacts), and (b) the build tool has read-only access and cannot modify anything.

This situation could be handled through folder- or artifact-level security rights. However, this situation occurs often enough that some customers have found it useful to create a view tailored to the needs of the build tool. Such a *build view* is often created as a read-only reference view based on a promotion state. The reference view may be rooted at a non-root folder of the parent view. Consequently, a limited set of artifacts are exposed, which cannot be modified. Whenever the promotion state is assigned to a new view label, the new artifact revisions automatically “appear” in the build view. That is, the build view follows changes to the promotion state.

Because security is easier to administer at the view level, a build view is often a more efficient way to accommodate build applications.

Proper Use of Views

Regardless of view type, below are some general do’s and don’ts to consider when creating and managing views:

- Try to ensure that items in the main view refer to the main (`1.n`) branches of its artifacts. Also, be careful not to delete the `1.n` branch of an artifact unless it has truly achieved end-of-life.
- *Leaf-most* views should be considered disposable. Eventually, when a view’s corresponding development activity is finished or its corresponding maintenance release is no longer supported, it should be deleted. This is important to prevent projects from growing unbounded, and it keeps the project structure uncluttered and easy to navigate. Deleting leaf views provides the **Server Administration** tool “purge” process with the maximum opportunity to actually shrink the database and vault by removing unneeded data.
- Three or four view levels in a project are normal. If you find your project has more than that, you might not be following best practices for views. For example, you may be making the mistake of performing new development work directly in the main view, forcing other active views to keep spawning child views. Or, you may be neglecting to promote the latest and greatest changes to the main view.
- Don’t use branch-none (“floating”) views except in extremely rare cases when you understand exactly how they work.
- You can “refactor” projects that have become too big by moving items in the main view to another project. Old change requests, tasks, and other projects can be moved to an “archive” project so they can still be accessed without cluttering the main view of an active project. If you have a project with numerous modules or applications, a large number of files can cause it to take too long to select “all descendants” in the main view. You can break the project up by creating new projects and moving (not sharing) folders and items corresponding to whole components or applications from the old project to the new projects. Do this when the main view reaches a major milestone such as after a new release. If you want to refactor the project by reorganizing the folder tree, do this before you split up the project. Create view labels before and after you refactor a project so you have markers for what changed.

Creating and Configuring Views

This topic provides the basic procedure for creating a new view based upon an existing view.

1. Display the project view upon which the new view will be based.
2. Choose **View** > **New**. The **New View Wizard** opens.
3. Select one of the available options from the **View Type** list:

   Branch All Will be based on a configuration of the currently open view. All items in the new view will be set to branch when they are modified. This branch behavior can be changed later for individual items in the new view so that changed items do not branch.

   Reference Allows users to read from and write to a subset of the parent view's current configuration. Any changes appear in both the reference view and its parent.

   Read-only Reference Allows users to read from a subset of the parent view. Unlike a read-write reference view, the contents of a read-only reference view may be current (floating) or configured to a point in the parent view's past by specifying a label, promotion state, or time.

4. Type a **Name** and a **Description** for the view in the appropriate fields and click **Next**.
5. Select the **Root Folder** for the new view and click **Next**.

   Note: The **New View Wizard** skips this step for a **Non-Derived** view.

6. Type or browse for the name of an appropriate **Default Working Folder**.

Caution: For a **Branch All** or a **Non-Derived** view, always use a working folder that is different from the one used by the parent view. Using the same working folder for the parent and child views can cause changes in one view to be overwritten when files are checked out from the other view. It can also result in incorrect or, at least, misleading file status indicators. For a **Reference** or **Read-Only Reference** view, you can use the same working folder as the parent view.

7. Click **Next** to display the **Select Types** page.
8. Select the item types to include in the new view. The new view will include items of the selected types from the parent view, unless the **Override default types** option has been selected in the **View Access Rights** dialog box.

   Tip: To create a view with no shared items, use the **Branch All** view type and clear all the check boxes on the **Select Types** page.
9. Click **Next** to display the **Configuration** page.

   If you are creating a **Non-derived** or **Reference** view, click **Finish**.

   Note: It is not necessary to display the **Configuration** page for a **Non-Derived** view because no items from the parent view are included. It is also not necessary to display the **Configuration** page for a **Reference** view because the items have the same configuration as those in the parent view.
10. Select one of the available configuration options on the **Configuration** page.
11. Click **Finish**.

View Configuration Options

The following describes the view configuration options available in the **New View Wizard**.

Floating Configuration

Not recommended. All the items in the new view will be identical to the corresponding items in the current parent view. Changes to an item in the parent view will be made to the corresponding item in the new view until that item branches, while changes to an item in the new view will be reflected in the parent until the item in the new view branches. (In many cases, the first change to that item will result in branching). New items in the parent view will appear in a branching view. However, new items added to a branching view will appear in the parent view only if the new view is the `Branch none` type.

Labeled Configuration

All the items in the new view will have had the specified label in the parent view. In all cases, the revision of the item to which the label was attached is the tip revision in the new view. This option is disabled if the parent view has no view labels. Changes to the parent view do not affect the new view, including changes to the label upon which the view is based. Unless an item is set to `Branch on change` in the new view, it will be read-only and you cannot change it.

Promotion State Configuration

All the items in the new view will have been part of the specified promotion state in the parent view. In all cases, the revision of the item that was part of the promotion state is the tip revision in the new view. This option is disabled if the parent view has no promotion states defined for it. Changes to the parent view will not affect the new view, including changes to the promotion state or its assigned label. Unless a specific item is set to `Branch on change` in the new reference view, it will be read-only and you cannot change it.

Configuration As Of

The new view will contain only the items that existed at the date and time you specify. In all cases, the tip revision of each item in the new view is the revision closest to, but before, the specified time. Changes to the parent view will not affect the new view. Unless a specific item is set to `Branch on change` in the new view, it will be read-only and you cannot change it.

View Type Options and Settings

The table below lists the settings that must be selected in the **New View Wizard** to create the different views.

Desired Characteristics for New View	Options to Set
Branching: **Branch All (Not Floating)** **New Items** New items in the child view do not appear in the parent view. New items in the parent view do not appear in the child view. **Existing Items** Existing items in the child view are the same as in parent view at the time of configuration, until the item in the child view branches. Changes cannot be made to an item in the child view if the change does not result in branching. **Item Behavior** **Branch on Change** check box is enabled and selected for all items that can branch. Any change to a child item that can branch results in the branching of that item, unless its **Branch on Change** check box has been cleared. After branching, the check box is disabled.	**View Type** `Branch All.` **Root Folder** Selected from parent view. **Working Folder** Should be different from that of the parent, to avoid conflicts. **Configuration** Other than floating (`Labeled`, `Promotion State`, or `As of specific date`). **Note: Addressed In Build** field when the child view is created, `Next Build` will not be replaced by a build label until the change request branches.
Read-only Reference View (Frozen) **New Items** New items cannot be added to child view. New items in parent view appear in child view. **Existing Items** Existing items are the same in the child view and the parent view; they can be changed only from the parent view. **Item Behavior** **Branch on Change** has the same setting as the parent item, but is irrelevant; no change can be made.	**View Type** `Read-only Reference`. **Root Folder** Selected from parent view. **Working Folder** Usually the same as that of the parent view. **Configuration** Other than floating (`Labeled`, `Promotion State`, or `As of specific date`). These views can be rolled back.
Non-derived View (also called **Blank Branching View**) **New Items** New items in child view do not appear in the parent view. New items in the parent view do no appear in the child view. **Existing Items** Existing items in the parent view do not appear in the child view. **Item Behavior** **Branch on Change** check box disabled.	**View Type** `Non-Derived`. **Root Folder** N/A. **Working Folder** Should be different from that of the parent to avoid conflicts. **Configuration** N/A.
Reference View (Also called **Read/Write Reference View**) **New Items** New items in the child view appear in both views. New items in the parent view appear in both views if they are in the subset accessed by the child view.	**View Type** `Reference`. **Root Folder** Selected from parent view. **Working Folder** Usually the same as that of the parent view. **Configuration** N/A. Always floats.

Desired Characteristics for New View	Options to Set
Existing Items Existing items are the same in the child view and the parent view. They can be changed from either view. **Item Behavior** **Branch on Change** has the same setting as the parent item. **Note:** Labels created and objects deleted in the child view appear and disappear in the parent view; this is not true for other types of child views.	
Branching: **Branch None (Not Floating)**. Not recommended. **New Items** New items in the child view do not appear in the parent view; new items in the parent view do not appear in the child view. **Existing Items** Existing items in the child view are the same as in the parent view at the time of configuration, until the item in the child view branches. Changes cannot be made to an item in the child view if that change does not result in branching. **Item Behavior** **Branch on Change** check box enabled, but initially cleared. No change to a child item that can branch results in branching until the Branch on Change check box is selected. After branching, the box is disabled.	**View Type** `Branch None` (an advanced type). **Root Folder** Selected from parent view. **Working Folder** Should be different from that of the parent, to avoid conflicts Configuration. **Configuration** Other than floating (`Labeled`, `Promotion State`, or `As of specific date`). **Note: Addressed In Build** field when the child view is created, `Next Build` will not be replaced by a build label until the change request branches.
Branching: **Branch None (Floating)**. Not recommended. Use only when a different set of view labels is needed for the same data. **New Items** New items in the child view appear in the parent view; new items in the parent view appear in the child view if they are in the subset accessed by the child view. In the child view, new items from the parent have the **Branch on Change** check box cleared. **Existing Items** Existing items are the same in the child view as in the parent view; they can be changed in either the parent or child view until the item in the child view branches. However, items deleted from one view are not deleted from the other. **Item Behavior** **Branch on Change** check box enabled, but initially cleared No change to a child item that can branch results in branching until the Branch on Change check box is selected. After branching, the check box is disabled.	**View Type** `Branch None` (an advanced type). **Root Folder** Selected from parent view. **Working Folder** Should be different from that of the parent to avoid conflicts. **Configuration** Floating. See the note below.
Branching: **Branch All (Floating)**. Not recommended. **New Items** New items in the child view do not appear in the parent view; new items in the	**View Type** `Branch All`, `Float` (an advanced type). **Root Folder** Selected from parent view.

Desired Characteristics for New View	Options to Set
parent view appear in both views if they are in the subset accessed by the child view. In the child view, new items from the parent have the **Branch on Change** check box selected. **Existing Items** Changes to existing items in the parent view appear in the child view until the corresponding item in the child view branches. Changes to existing items in the child view can appear in the parent view, but only if the **Branch on Change** check box for that item is cleared. However, items deleted from one view are not deleted from the other. **Item Behavior** **Branch on Change** check box is enabled and initially selected for all items that can branch. Any change to a child item that can branch results in the branching of that item, unless its Branch on Change check box has been cleared. After branching, the box is disabled.	**Working Folder** Should be different from that of the parent to avoid conflicts. **Configuration** Floating See the note below.
Branching: **Branch All (Floating)** **New Items** New items in the child view do not appear in the parent view. New items in the parent view appear in both views if they are in the subset accessed by the child view. In the child view, new items from the parent have the **Branch on Change** check box selected. **Existing Items** Changes to existing items in the parent view appear in the child view until the corresponding item in the child view branches. Changes to existing items in the child view can appear in the parent view, but only if the **Branch on Change** check box for that item is cleared. However, items deleted from one view are not deleted from the other. **Item Behavior** **Branch on Change** check box is enabled and selected for all items that can branch. Any change to a child item that can branch results in the branching of that item, unless its **Branch on Change** check box has been cleared. After branching, the box is disabled.	**View Type** `Branch All`. **Root Folder** Selected from parent view. **Working Folder** Should be different from that of the parent to avoid conflicts. **Configuration** Floating See the note below.

Note: If users are likely to perform many move and share operations, using branching, floating views can result in multiple unwanted references to the same folders or items, causing confusion. Also, if a change request has `Next Build` in the **Addressed In Build** field when the child view is created, `Next Build` will be replaced by the parent's next build label, unless the change request is first branched.

Change Management within a View

Undoubtedly, your most active views will be activity and sandbox views. But, unless you've figured out how to write defect-free software, you'll also need to make changes to release views. And, although we don't recommend it as a general practice, for small changes, it often makes sense to perform updates directly in the main view. In all of these cases, your team has several choices for the process by which they perform updates.

The appropriate process depends on the size of your development team (which affects the probability of conflicts) and the level of detail with which you need to track changes. So, let's first review the ways in which changes are documented, tracked, and queried in StarTeam:

- Each new artifact revision can receive a *revision comment* where the user can document why the revision was created. Projects can be configured to require the revision comment for files.
- An artifact's revisions can be reviewed and compared in the **History** tab of the StarTeam Cross-Platform Client. Historic revisions include the revision comment, modifying user, modification timestamp, version number, and all other properties as of that revision. The StarTeam Cross-Platform Client allows you to customize what properties are shown in the **History** tab and to compare file content changes in a dedicated **File Compare** pane.
- Unless audit generation has been disabled for the configuration, an audit record is generated for both revision-generating changes and other updates such as moving an artifact to a new folder or attaching an artifact to a label. You can query audit records in the StarTeam Cross-Platform Client's **Audit** tab. The length of time that audit records are retained is configurable via the **Server Administration** tool. We recommend that you always enable audit generation and retain records for at least 90 days.
- Through process items or explicit actions, links can be used to connect change items such as CRs and tasks to files that were added and modified on their behalf. Links generated via process items are pinned to the specific artifact revisions involved at both ends. You can view an artifact's links in the StarTeam Cross-Platform Client's **Link** tab. To navigate to a linked item, right-click the item and choose **Select Linked Item** from the shortcut menu.
- Artifacts can be attached to revision and view labels so that revisions related to a common task or milestone are easily identified. (StarTeam can automatically attach new and modified files to a revision label at add/check-in time.) Choose **Select** > **By Label** to see the artifacts attached to a label. The labels to which each artifact revision is attached can be seen via the StarTeam Cross-Platform Client's **Label** tab.
- TheStarTeam Cross-Platform Client offers several text-based reports and graphical charts for reporting on historic revisions, links, audit records, and other information. You can also create custom report templates that can be saved and reused.
- All of the change information described above can be accessed via the StarTeam SDK. If none of the built in reporting mechanisms are sufficient for your needs, you can build custom reporting tools or applications. Custom tools can even be centrally stored and automatically downloaded to StarTeam Cross-Platform Client users, where they can be activated via the toolbar.
- The Datamart product allows information from a StarTeam project, configuration, or multiple configurations to be extracted and loaded into a relational database, which can be queried and analyzed through reporting and business information tools.

Now that we've looked at the primary means by which you can report on changes within a view, let's look at ways in which you can manage change. We'll do this by examining three scenarios in increasing order of "formality".

Scenario 1: Working in a Small Team

If you're working on a project managed by a small team, you may not need much formal change management. Furthermore, if code modules are clearly assigned to individuals, or if you can just lean out your office door and yell, "Hey, I'm working on the report module", then the possibility for conflicts may be very low . In this case, your change process may be very simple:

- Maintain a set of local working folders where you keep the latest files for each view.
- Before you begin a new development task, get your working folders up-to-date by checking out any “Missing” or “Out of Date” files.
- As you modify files in your working folders, you’ll see their status in the **File** tab become `Modified`. New files will appear as `Not in View` because they have not been added to the view yet. New folders will appear as `Not in View` in the **Folder** tab. If you enable **Show Not-In-View Folders** in the **Folder Tree** menu, you’ll see not-in-view folders in the folder tree regardless of which tab is selected. The folder tree always shows `Current` folders and those `Missing` from your workspace.
- When you’re ready to check-in, refresh your window (if you don’t have automatic refresh turned on) to see if another developer added or modified anything that may affect you. Check-out any new `Missing` or `Out of Date` files.
- If another developer checked in a file that you modified locally, its status will become `Merge`. The best way to resolve the merge conflict is to check out the file and answer **Yes** to the **Do you want to merge** prompt. This will launch the **File Compare Merge** tool. If there are no conflicts, the merge tool usually just saves the merged result file over your existing work file. Otherwise, you may be prompted to review and tweak the result file before continuing. Once saved, the file’s status will change to `Modified`.
- When you have only `Not in View` and `Modified` files, to do a build, run unit tests, and/or do any other validation steps your team requires.
- Finally, commit your changes to the view. You can select all of the new (`Not in View`) and modified files and check them all in at once. If you add/check-in multiple files in a single step, they are committed as an atomic transaction. Moreover, file content is pushed to the server “outside” of transaction state in a restartable manner. So, if you cancel a large multi-file add/check-in operation before it commits, you can restart it by just doing it again, only files that didn’t make it in the first attempt will be sent to the StarTeam Server. The commit happens at the end only when all content has arrived using a single transaction.

Note: One caveat of adding new files and checking-in modified files in the same check-in dialog is that the revision comment is also used to initialize the **Description** property of new files. If you want accurate descriptions for new files, add each one separately or modify each file’s description after adding it. New folders are added automatically, you only need to explicitly add new folders on the server when they are empty on the client.

And that’s it. You get revision history and audit records automatically. This process doesn’t fool with labels, and you don’t get process item links either. But it might be all that your organization needs.

Scenario 2: Preventing Merge Conflicts

As your team size grows, the problem you may find with the previous scenario is encountering too many unexpected merge conflicts. Your team may want everyone to know when you’re about to make changes to a file before you actually do so. To better communicate intent and reduce the likelihood of merge conflicts, you can amend the process outlined in the previous scenario as follows:

- In the **Personal Options** dialog box, select the options **Mark unlocked working files as read-only** and **Clear file locks on check-in**.
- When you want to edit a file, lock it via the StarTeam Cross-Platform Client first.
- When you check-in modified files, the tools will automatically unlock them for you (in the same transaction in which the changes are committed).

This process both informs team members of who is working on what, and it minimizes merge conflicts. Note that your team can require this process by setting the project-level options **Require exclusive lock when files are checked-in** and **Mark unlocked files read-only**. It’s up to you and your team to decide if you want this level of enforcement.

Scenario 3: Using Process Items

The next step in a more formal intra-view change management practice is to use process items so that all file modifications are linked to an appropriate change item. You can enforce the use of process items at the project level using the **Process Rules** options. These options let you:

- Require the selection of a process item when new files are added or modified files are checked in.
- Select which item types are eligible as process items. Your choices are change requests, tasks, and requirements.
- Specify whether or not the **Status** of each item type will be considered in order to be used as a process item and which status values are permitted. For example, you can specify that change requests must have a status of `Open`, but tasks can be used regardless of status.

When you enforce process items, existing files cannot be modified and new files cannot be added until an eligible process item is selected. Consequently, new and modified files are automatically linked to the selected process item, enhancing the context information of these changes. Moreover, the links are created in the same atomic transaction in which the file updates are performed.

Note: In the add and check-in dialogs, you can elect to mark the selected process item `Closed` (change requests), `Finished` (requirements), or `Complete` (tasks). If you choose this option, the process item update is also performed in the same atomic transaction.

Switching Views

When you open a project, you select a view of that project. Once a project is open, however, you can switch to another view. By default, the newly selected view always opens with the Current configuration, regardless of the configuration it had when you last exited it.

1. Click **View** > **Select View**. The **Select a View** dialog box opens, which shows the views hierarchically.
2. Check **Open in New Window** if you would like to keep your existing view open rather than changing the window to the new view. When you check this item, a new view window will open in the Cross-Platform Client window and the name of the new view will be added to the list of opened views on the **Window** menu.
3. Select a name from the **View** list and click **OK**, or simply double-click the view name to open your project in that view.

If you do not wish to use the default configuration, you can roll back the current view configuration.

Note: If you wish to open two different StarTeam view windows in the same project at the same time, select **Project** > **Open** to display the additional view.

Changing a View's Default and Alternate Working Folders

Make sure that everyone is logged off from the server and that the server is locked before you change the *Default Working Folder*. It is just as critical to perform these actions as it is when you change custom fields or do anything else that affects all users.

When you change the **Default Working Folder**, not only the path to the working folder but the path to each child folder in the view may be similarly modified—not just for you, but for everyone working with that view.

Caution: Do not change the **Default Working Folder** unless you are a project administrator. These default settings affect all users and incorrect settings cause other users to be unable to check out StarTeam files. The default settings should only be set to the name of the folder. If you want to use a different location for your working folder than the **Default Working Folder** path, specify an **Alternate Working Folder** path.

1. Choose **View** > **Properties** to open the **View Properties** dialog box.
2. Click the **Name** tab.
3. Do one of the following:
    - Select **Alternate** to create a different working folder for only yourself.

- If you are a project administrator, select **Default** to specify the default repository path for all users.

4. Type the name of a new working folder or browse for a path to a working folder. If you browse for the path, it becomes an absolute path. This path can be edited, however, to enable you to work on a computer that uses a different letter for its hard drive.

   Note: It is important that the **Default Working Folder** point to a location that is physically discrete for each user, such as a drive on that user's workstation or a personal directory on a shared file server.

Deleting Views

Before deleting a project view, be absolutely certain that you wish to do so. After performing this operation, you will no longer be able to access any item in the view that is not shared with another project or view. Deleted views are also not visible in the **Select View** dialog box, although deleting a view does not remove any data from the server database.

If other users are connected to the project view when it is deleted, they will receive a `Deleted` message the next time they initiate a view command.

Note: A view cannot be deleted if it has derived child views.

1. Choose **View** > **Delete** .
2. A message box appears asking you to confirm.
3. Click **Yes**. A confirmation dialog box asks you to type the name of the project.
4. Type the project name, which is case-sensitive, in the **View Name** field.
5. Click **OK**.

Modifying View Names or Descriptions

Root views initially are assigned the same name as the project. If you have the appropriate privileges, however, you can change the name and/or description of the root view or its children.

1. Click **View** > **Properties**. The **View Properties** dialog box displays.
2. On the **Name** page, type a new name in the **Name** field.
3. Type a new description in the **Description** field.
4. Click **OK**.

Refreshing Views

You can refresh data in the StarTeam project view window in several different ways, depending upon what you wish to update:

To refresh a view

From the **View** window, do one of the following:

- Press **F5** to refresh the upper right pane for the current item list, such as the **File** list or **Change Request** list.
- Press **Ctrl+F5** to refresh the upper pane and simultaneously collapse all open groups.
- Press **Shift + F5** to refresh the entire view (all item lists in all tabs as well as the folder hierarchy).

Tip: You can turn on **Auto Refresh** in the **Personal Options** dialog box on either the **Workspace** tab or the **StarTeamMPX** tab. **Auto Refresh** is designed to perform even if the window is minimized.

Reviewing or Modifying View Properties

Sometimes you may want to look at the values and properties originally used to create a view. Reviewing this information may help you understand the behavior of changes within the view or the views that have been derived from it. Also, if you have access rights to do so, you may be able to modify view properties as well.

1. Choose **View** > **Properties**. The View **Properties** dialog box opens.
2. Select the **Name** tab to see or change the following:
   - Name and description of the view
   - Who created the view and when it was created (read-only)
   - Whether the items are set to branch on change
   - Whether a central or a per folder repository is being used
   - Working folder path

   Note: Your access rights determine which items you can change.
3. Select the **Hierarchy** tab to review the list of views for this project and their relationship to one another.
4. Select the **Type** tab to see:
   - View type.
   - Whether the view is a root, branching, non-derived, or a reference view.
   - For branching views, whether the original default was Branch All or Branch None.
   - Parent view on which this view was based.
   - Parent configuration used to create this view.
5. Click **OK**.

Rolling Back the Current View Configuration

By default, a view has a current configuration – that is, it displays the latest revisions of the items in the project. However, you can roll back a view to a past state based on a label, promotion state, or a point in time. Note that rolling back a view in this way configures it for the current user only.

When you roll back a view, this action prevents it from changing, until you select **Current Configuration** or close the project, which automatically changes the view to Current. You cannot check in files, update change requests, and so on in a rolled-back view because you cannot change the past.

When you configure a view, you can base it on a promotion state. Whether the state is assigned to **Current** or to a specific view label, any view configured to a promotion state is read-only. This read-only status can create problems for a user who needs to, for example, both look at files as they existed earlier and create new change requests. Because CRs can only be added to a current configuration, the user may need to have two windows open for the view: one configured to a point in the past and one set to the current configuration.

Rolling Back a Current View

1. Choose **View** > **Select Configuration** . The **Select A View Configuration** dialog box opens.
2. Select a view configuration option:

   Labeled Configuration All the items in the new view will have had the specified label in the parent view. In all cases, the revision of the item to which the label was attached is the tip revision in the new view. This option is disabled if the parent view has no view labels. Changes to the parent view do not affect the new view, including changes to the label upon which the view is based. Unless an item is set to `Branch on change` in the new view, it will be read-only and you cannot change it.

Promotion State Configuration All the items in the new view will have been part of the specified promotion state in the parent view. In all cases, the revision of the item that was part of the promotion state is the tip revision in the new view. This option is disabled if the parent view has no promotion states defined for it. Changes to the parent view will not affect the new view, including changes to the promotion state or its assigned label. Unless a specific item is set to `Branch on change` in the new reference view, it will be read-only and you cannot change it.

Configuration As Of The new view will contain only the items that existed at the date and time you specify. In all cases, the tip revision of each item in the new view is the revision closest to, but before, the specified time. Changes to the parent view will not affect the new view. Unless a specific item is set to `Branch on change` in the new view, it will be read-only and you cannot change it.

3. Click **OK**.

Returning to the Current Configuration

1. Choose **View** > **Select Configuration** . The **Select A View Configuration** dialog box opens.
2. Select **Current Configuration**
3. Click **OK**.

Basing a View Configuration on a Promotion State

1. Choose **View** > **Select Configuration** . The **Select A View Configuration** dialog box opens.
2. Select **Promotion State Configuration**.
3. Select a state from the list. Configuring a view to a promotion state maintains the dynamic nature of the promotion state. If the promotion state view label changes, the view is configured to the new view label on the next refresh. The promotion state name appears on the status bar. If you exit the view and return, you return to the current configuration.
4. Click **OK**.

Granting View-Level Access Rights

Usually, granting access rights at the project level is not a fine enough level of granularity. For example, one set of developers may maintain Release 1.0 of the product in one view, while another set of developers writes the source code for Release 2.0 in another view.

To handle this situation, you may want to create new groups, such as 1.0 Developers, 2.0 Developers, 1.0 Testers, and 2.0 Testers. You can give the 1.0 Developers and 1.0 Testers access to files and/or change requests in the Release 1.0 view and. Then you can give the 2.0 Developers and 2.0 Testers access to files and/or change requests in the Release 2.0 view.

In this case, you need to set access rights at the view level. However, you must still set project access rights at the project level because that is the only place where the Project node appears.

View and Child View Access Rights Access rights in a child view at the view level are independent of the access rights of the parent view at the view level. Therefore, a child view starts out with no access rights at the view level.

A new child view is represented by a different object in the repository from the parent view. It has a different name, description, place in the view hierarchy, etc.

View-level access rights can be set for a new child view. For example, suppose a reference view contains only one branch of the parent view's folder hierarchy. The reference view has a root folder named QA Tests. In this situation, you can make the **Testers** the only group with file access rights in the reference view, even if **Developers** is the only group that has file access rights in the parent view.

Access Rights at Different Levels

Sometimes a group has different access rights at the view and the project levels for the same type of object in the same view. In this situation, the access rights at the lowest level are enforced.

When the StarTeam Server searches for access rights, it starts from the lowest level and moves to the highest level. When it finds a level at which a group has access rights, it does not search any higher levels for that type of object.

Remember that the project access rights exist only at the project level, so the project level is always searched for these rights. File access rights, on the other hand, exist at the file, folder, view, and project levels. the server stops at the first level at which it finds file access rights.

View Access Rights

When you select the **View** > **Access Rights** command to open the **View Access Rights** dialog box, the rights shown are for the current view. The rights available from the **View** node are also available from the **View** node in the **Project Access Rights** dialog box. In the latter case, the rights cover all views in the project rather than an individual view. It also include a container-level right that allows users or groups to create views for the project. This right is not available on the **View** node of the **View Access Rights** dialog box.

The following table describes the access rights that are available from the **View** node in the **Project Access Rights** dialog box. Most of these access rights also appear on the **View** node of the **View Access Rights** dialog box, but apply only to the current view.

Generic Object Rights

See object and its properties	Change view properties. View properties that can be modified are the view's name, description, working folder (also the root folder's working folder), branch setting for shared items, and file status repository setting.
Modify properties	Modifies the view properties.
Delete object	Deletes the object from the view.
Change object access rights	Changes the access rights of the selected object in the view.

View-Specific Rights

Create view labels	Creates view labels. These labels will be automatically attached to the folders and items in the view. Users with this right but not the right to attach labels can still create labels.
Modify view labels	Changes the properties of view labels. For example, this right allows a user to freeze labels so that they cannot be adjusted
Delete view labels	Deletes view labels. This action automatically detaches the view labels from the folders and items that had the labels. Users with this right but not the right to detach view labels can still delete view labels.

Create revision labels	Creates revision labels. Users with this right but not the right to attach labels can still create labels.
Modify revision labels	Changes the properties of revision labels. For example, this right allows a user to freeze labels so that they cannot be adjusted.
Delete revision labels	Deletes revision labels. This action automatically detaches the labels from the folders and items that had those labels. Users with this right but not the right to detach revision labels can still delete revision labels.
Define promotion model	Creates, deletes, and reorders promotion states and edit their properties. After creating a promotion state, you must exit and reenter the Promotion dialog if you want to set access rights for the newly created state.
Override default types	Overrides default types.

Generic object container rights

Create views	Creates views in the current project. This container-level right is available only when you select the View node from the Project Access Rights dialog.
Override default types	Allows users to override the default set of types included when a new view is created.

References Overview

You can base a folder or item in one application location on another folder or item stored in a different location within the same server configuration. **References** indicate the relationships between an original folder or item and the others based on it. References can be used to decide whether the changes you have made to a folder or item in one location need to be applied elsewhere.

Found in the lower pane of the clients, the **Reference tab** shows the relationships between the selected item and other folders or items with which it is associated. A folder or item may be associated with more than one project, view, or parent folder in the same server configuration because of sharing or because a child view has been created. Each instance of the original folder or item has a reference. Item references (including folders) can be viewed on the Reference tab of the lower pane. You can also view folder references from the **Folder** tree in the left pane by selecting **Advanced** > **References** from the **Folder** tree or context menu to display a dialog.

Understanding References

StarTeam creates at least one reference to a folder or item whenever:

- You create or add a folder or item.
- A branching child view is created that will contain that folder or item. As a branching child view is created from its parent, a subset of the folders or items in the parent becomes part of the child view. StarTeam automatically shares the folders and items in that subset into the child view.
- You manually share a folder or item from one location to another.

As you add, share, or move a folder or item, more than one reference to it may occur if the view is a child view that branches and floats, or if the view has child views that branch and float.

Actions Causing StarTeam to Create References

For example, suppose you want to move a file from one folder to another in the same view. Suppose that the view has two child views, both of which contain the file. That means that there are at least three references to this file, one in each of three views. Now, you move the file to another folder in the same view. The reference in the current view is moved to represent the new location of the file. Depending on the properties of the two child views, a new reference may be created for the file in each of the child views. The references in those child views to the file in its original location still exist, because the application does not assume that you want to change those references just because you have moved the file in the current view. You may end up with five references to this file that formerly had three references.

Note: Most administrators avoid branching, floating views if users are likely to perform many operations that result in additional references. For example, moving and sharing can result in multiple unwanted references to the same folders or items, which can cause confusion.

The following table explains what references StarTeam creates in the current view, the recipient view, the parent of the recipient view, and the children of the recipient view. This is often recursive. For example, if a reference is created in the parent view, new references might be created in the other children of that view or in the parent of that view, and so on, depending on what views are floating.

When a folder or item is...	...is a reference added to the view of the recipient?	...is a reference added to the parent view of the recipient view?	...is a reference added to the child views of the recipient view?
Part of a newly-created view	Yes, unless the new view is a reference view. In this case, a new view is not really being created, because a reference view is just a new way of looking at an existing view.) There is one reference for the folder or item in the newly-created view.	No, because the parent view is the source of the folder or item, so the reference in the parent view already exists.	No, because the newly-created view has no child views.
Added to the current view	Yes, there is one reference for the new folder or item in the current view.	Yes, if the current view is a branch none, floating child of the parent view. Otherwise, no.	Yes, if the child view is a branching (either branch none or branch all), floating child of the current view. Otherwise, no.
Shared within the current view	Yes, a new reference is created for the shared folder or item in the new location in the current view.	Yes, if the current view is a branch none, floating child of the parent view. Otherwise, no.	Yes, if the child view is a branching (either branch none or branch all), floating child of the current view. Otherwise, no.
Moved within the current view	No, the original reference is updated to reflect the move.	Yes, if the current view is a branch none, floating child of the parent view. Otherwise, no.	Yes, if the child view is a branching (either branch none or branch all), floating child of the current view. Otherwise, no.

How to View References

Consider the following example of four references that would display in the **Folder References** dialog box:

- `Help Files::Help Files::Help Files::starteam,1.0`
- `Help Files::Help Files\Freeze Check::Help Files\release 4\starteam, 1.0`
- `Help Files::Help Files\Freeze Check\New View::\Help Files\release 4\starteam, 1.0`

- `Help Files::Help files\variant 2::Help Files\release 4\starteam, 1.0.1.2`

In the above example, the selected folder has four references.

The Current icon indicates which reference represents the currently selected folder or item. Otherwise, this dialog contains the same information regardless of the view in which you selected the folder.

Each reference shows the following, separated by double colons (**::**):

- The project name (for example, Help Files).
- The path from the root view to the view containing the folder (or item). For example, `Help Files\Freeze Check\New View`, where `Help Files` is the name of the root view, `Freeze Check` is a child of the root view, and `New View` is a child of the `Freeze Check` view.
- The path to the folder within the view. In the case of an item, the path is to the parent folder of the item.
- In the case of an item, the name or number associated with that item. This can be the filename, change request number, the requirement number, the task number, or topic number.
- The tip revision number for the folder (or item) in that view. (This information is separated from the rest of the reference by a comma, rather than the double colon.) For example, the folder in the example is revision `1.0` in all views except for the `variant 2` view (see the last leaf in the example tree). In the `variant 2` view, the revision number for the folder is `1.0.1.2` which indicates that the folder has been branched from the `1.0` revision in its parent view and has had three revisions in the `variant 2` view. Those revisions are `1.0.1.0`, `1.0.1.1`, and `1.0.1.2`.

In this example, the name of the project, the name of the root view, and the root folder in the root view all have the same name.

You can resize the **Folder References** dialog box (by dragging an edge or corner). It displays scroll bars when appropriate. The references in bold indicate which revisions of the currently selected folder or item are its descendants. In other words, the currently selected folder or item is part of the revision history for the references that are in bold.

Consider the following example from the Reference tab shows the references for a file (`AUDITSCC.DOC`). The reference for the currently selected file indicates that revision of the file is `1.6`. As indicated by the bolding of its reference, revision `1.8` is the only descendant of revision `1.6`. If a defect is found in revision `1.6` of `AUDITSCC.DOC`, the bolding helps you determine which descendants of `1.6` may also need the corrected lines. In this case, you may only new to update `1.8`.

- ⊟ `Help Files::Help Files::Help Files\starteamp::AUDITSCC.DOC, 1.8`
- `Help Files::Help Files\Freeze Check::Help Files\starteamp::AUDITSCC.DOC, 1.1`
- `Help Files::Help Files\Freeze Check::New View2::starteamp::AUDITSCC.DOC, 1.1.1.0`
- `Help Files::Help Files\varc::Help Files\starteamp::AUDITSCC.DOC, 1.6`
- `Help Files::Help Files\variant 2::starteamp::AUDITSCC.DOC, 1.2`

Initial References

When you add a folder or item to the application, StarTeam creates a reference. Consider the following example of a folder hierarchy for a newly-created project. In this example, it is the folder hierarchy for the root view of that project.

Before you make any changes to the folder properties of the `Source Code` folder, the **Folder References** dialog box would contain exactly one reference to it. For example, `Big Product::Big Product::Big Product\Source Code, 1.0`.

As you make changes to the folder properties of the `Source Code` folder, the revision number might change from 1.0 to 1.1 and later 1.2. However, there will still be only one reference to this folder in the **Folder Reference** dialog box.

If a reference view is created (to be used, for example, by a group of reviewers), the view hierarchy for the `Big Product` project would contain two views, but the `Source Code` folder would continue to have just one reference. A Reference view contains a subset of the folders in its parent view, but those folders are the same folders as those in the parent view. They cannot branch. For example, after creating a reference view for reviewers, the Folder References dialog box would contain the following information:

Big Product::Big Product::Big Product\Source Code, 1.2.

Once you share the folder manually or share it automatically when you create a branching child view, additional references then display in the **Folder References** dialog box.

Tip: Do not confuse reference views with folder and item references. A reference view looks like a new view, but it is really a subset of an existing view. A folder or item reference is like a reference count. It indicates how many copies of the object exist or can exist if the object branches in each of its new locations. The creation of a reference view does not result in the creation of any folder or item references.

References Created by Branching Views

When you create a branching view, each folder or item automatically shared from the parent view to the child view acquires an additional reference. In the view hierarchy (which you can display from **View** > **Select View**), the new reference is a child of the original reference.

Folder References Created by Branching Views

Suppose that when the 1.0 version of Big Product ships, the team leader creates a branching view (based on the ship date for the 1.0 version) to be used for service packs, while new development on version 2.0 still continues in the project root view. These actions would result in the following view hierarchy:

- `Big Product`
- `  Big Product 1.0 Plus Service Packs`
- `  Reference view for reviewers`

At this point, two references display in the **Folder References** dialog box. When you are in the root view, `Big Product`, the `Folder References` dialog box for the `Source Code` folder contains the following information:

- `Big Product::Big Product::Big Product\Source Code, 1.2`
- `Big Product::Big Product\Big Product 1.0 Plus Service Packs::Big Product\Source Code, 1.2`

When you are in the child view, `Big Product 1.0 Plus Service Packs`, the **Folder References** dialog box for the `Source Code` folder contains the following information:

- `Big Product::Big Product::Big Product\Source Code, 1.2`
- `Big Product::Big Product\Big Product 1.0 Plus Service Packs::Big Product\Source Code, 1.2`

The Current (You Are Here) icon indicates which reference represents the currently selected folder or item. Otherwise, this dialog contains the same information regardless of the view in which you selected the folder. StarTeam indents the reference for a child view beneath the reference for its parent. The references in bold indicate which revisions of the folder or item are descendants of the folder or item with the Current (You Are Here) icon. In other words, the current folder or item is part of the history for the revisions that are in bold.

In the previous two examples, both references were represented in bold text. In the next example, this is not the case. This is because the properties of the *Source Code* folders in both the parent view and the child view have changed. The folder for the parent has revision `1.3`, and the folder for the child has revision `1.2.1.0`. Both folder histories have gone in different directions.

- `Big Product::Big Product::Big Product\Source Code, 1.3`
- `Big Product::BigProduct\Big Product 1.0 Plus Service Packs::Big Product\Source Code, 1.2.1.0`

The current folder is a descendant of itself, so it is always represented in bold text. However, it has evolved from the parent folder, so it is no longer in the history of the current folder. Accordingly, the **Folder References** dialog box for the parent folder would present the following information:

- `Big Product::Big Product::Big Product\Source Code, 1.3`
- `Big Product::BigProduct\Big Product 1.0 Plus Service Packs::Big Product\Source Code, 1.2.1.0`

File References Created by Branching Views

When you look at the history of a folder or item, you see its ancestors, not its descendants. However, if you change the tip revision in one location and that revision is an ancestor of the tip revision in another location, you might also want to apply your change to the tip revision in the other location (the object of the first descendant). The way to tell if a revision has descendants is to look at its references. Consider the following example showing the references for a file (`AUDITSCC.DOC`):

- `Help Files::Help Files::Help Files\starteamp::AUDITSCC.DOC, 1.8`
- `Help Files::Help Files\Freeze Check::Help Files\starteamp::AUDITSCC.DOC, 1.1`
- `Help Files::Help Files\Freeze Check::New View2::starteamp::AUDITSCC.DOC, 1.1.1.0`
- `Help Files::Help Files\varc::Help Files\starteamp::AUDITSCC.DOC, 1.6`
- `Help Files::Help Files\variant 2::starteamp::AUDITSCC.DOC, 1.2`

As the bold text indicates, if the current revision is `1.6`, then `1.8` is its only descendant. This also means that you would find revision `1.6` in the history for *1.8*.

If a defect is found in revision `1.6` of `AUDITSCC.DOC`, the bold text helps you determine the descendants of `1.6` that may also need the corrected lines. In this case, `1.8` may need to be updated. The other references are for revisions of the file that:

- Have already diverged (branched) and may be quite different than the current file.
- Are ancestors of the current file and less likely to need a change. For example, they may be in views that are read-only or no longer in use. Whatever the reason for the gap, the ancestors might require far more work than the changes you are about to check in.

You should check for descendants before (and perhaps after) you create a new revision of a folder or item. Before the change becomes a new revision in the application, you can see the descendants. Afterwards, you may see what other references have the same revision number as the newly-changed folder or item. If they, too, have the new revision number, then they, too, already have the new change. For example, the file may be floating in other views.

References Created by Adding Items to Views

The addition of a new folder or item to a parent or child view can result in one or two references, depending on the relationship between the two views.

If the child view is a branching, floating view, StarTeam creates a reference in each view when a new folder or item is added to the parent.

If the child view is a branching, floating view created using the **Branch None** option, StarTeam creates a reference in each view when a new folder or item is added to the child.

Floating Down in the View Hierarchy

When a view has a branching child view (whether created with the **Branch None** or **Branch All** option) and the child view is floating, any folder or item added to the parent view becomes visible in both views. The history of the folder or item indicates the view in which the object was created, and the reference hierarchy displays the reference that identifies the parent view as the parent reference.

For example, if a file you add a file to the parent view, its history in either view shows the name of the parent view—until the file branches in the child view.

The following table shows the history in the parent view for a file that was added to the parent view and floated downwards.

Tip: You can review historical item information using the **History** tab in the client.

View	Revision	Branch Revision
Big Product	2	1.1
Big Product	1	1.0

The following table shows the history in the child view for a file that was added to the parent view and floated downwards. The history of the file displays the name of the view from which the file was originally added to the application, until the file branches. Then it displays the name of the view in which the file branched.

View	Revision	Branch Revision
branch none floating	3	1.1.1.0
Big Product	2	1.1
Big Product	1	1.0

If you were to display the References tab for this file (`marketshares.doc`) after it has branched in the child view, you would see the following information:

- `Big Product::Big Product::Big Product\Marketing Documents::marketshares.doc, 1.1`
- `Big Product::Big Product\branch none floating::Big Product\Marketing Documents::marketshares.doc, 1.1.1.0`

Notice that the history clearly shows the parent view as `Big Product` before the file branches. The history and references for folders and items added to the parent view are similar to those for folders and items that were in the parent view at the time the child view was created.

Note: The name of the views in these examples makes the information easier to understand. You would probably never name a view parent or any other of the names shown these examples.

Floating Up in the View Hierarchy

When a view has a branching child view (created with the **Branch None** option) and the child view is floating, any folder or item added to the child view becomes visible in both views. This is not true of branching, floating child views that were created using the **Branch All** option.

The history of the folder or item indicates the view in which the object was created, but the reference hierarchy always displays the reference that identifies the parent view as the parent reference.

The following table shows the history in the parent view for a file that was added to a child view and floated upwards. Notice that, even though this is the history in the parent view, the history displays the name of the view from which the file was originally added to the application.

View	Revision	Branch Revision
branch none floating	3	1.2
branch none floating	2	1.1
branch none floating	1	1.0

The following table shows the history in the child view for a file that was added to the child view and floated upwards. The history of the file displays the name of the view from which the file was originally added to the application—until the file branches. Then it displays the name of the view in which the file branched. In this case, those two views just happen to be the same view.

View	Revision	Branch Revision
branch none floating	3	1.1.1.0
branch none floating	2	1.1
branch none floating	1	1.0

When you view the reference hierarchy for a file that floats upwards, you cannot tell that the file was added to the application from the branching child view, and you must investigate the history (using the History tab) of the file to determine where the file originated. For example, the Reference tab would contain the following reference hierarchy for the `slant.doc` file that floated upwards:

- `Big Product::Big Product::Big Product\Source Code\Timeout::slant.doc, 1.2`
- `Big Product::Big Product\branch none floating::Big Product\Source Code\Timeout::slant.doc, 1.1.1.0`

Floating Up and Down in the View Hierarchy

If the view hierarchy is deep (the root view has grandchildren, great-grandchildren, and so on), the use of branching, floating views can cause a great deal of confusion. For example, suppose you add a file to a grandchild of the root view. Further, suppose that this grandchild view was created using the **Branch None** option and that its parent (a child of the root view) was created using the **Branch None** option. The file you add can float up to the parent and grandparent of the current view from which it will, in turn, float back down to the current view. This results in:

- One reference to the file in the current view
- One reference to the file in the parent of the current view (the result of floating up from the current view)
- One reference to the file in the root view (the result of floating up from the parent of the current view)

More references are created if the current view has floating children, grandchildren, and so on. Still more are created if the root view or parent view have other floating children besides the ones mentioned above.

References Created by Manually Sharing Objects

As you share a folder or item from one location to another (whether in the same view or a different one) an additional reference is created for that object in the new location. The reference for the new folder or item becomes a child of the reference from the folder or item that was shared.

Reference Hierarchy Example for a Manually Shared File

The following example shows two references for a file named `timeout.cpp`. The file was manually shared from a folder named `Source Code` to a folder named `Timeout` in the same view. Notice that the second reference is based on the first, but created as a by-product of creating a branching view.

- Big Product::Big Product::Big Product\Source Code::timeout.cpp, 1.0
- Big Product::Big Product::Big Product\Source Code::Timeout::timeout.cpp, 1.0

The application does not differentiate between references based on what caused them to be created. However, you can tell from the hierarchy that the first reference is the source of the second reference, because the second reference is indented under the first. You can also tell, because they are in the same view, that a manual share or move occurred. (The second reference would be in a different view if it was created automatically when a child view was created.

A shared folder or item can branch, but may never do so. Regardless, some subset of its history is part of the history of the original folder or item.

Floating Up and Down in the View Hierarchy

If the view hierarchy is deep (the root view has grandchildren, great-grandchildren, and so on), the use of branching, floating views can cause a great deal of confusion. Suppose all the views except the root view branch and float. At its new location, depending on how views were created, the folder or item you share can float:

- Up the view hierarchy from the recipient view to the root view
- Down to all the recipient children of the view, grandchildren, and so on
- From the recipient view's parents, grandparents, and so on, to all of their other children

This can result in a reference to the folder or item in the new location in every view in the project's view hierarchy. Many of those views may have already had a reference to the folder or item in its old location.

The following example shows all the references created by sharing a file named `shared within child view.doc` from one location in the `branch none floating` view to another location in that same view. The first three references are the references that existed prior to the sharing operation. The fourth reference is the new reference in the root folder. It is shown as a child of the first location in the `branch none floating` view because it floated up from that view. The fifth and sixth references resulted from references that floated down to the `branch none floating` child view of that view.

- Big Product::Big Product::Big Product\Online Help::shared within child view.doc, 1.0
- Big Product::Big Product\branch none floating::Big Product\Online Help::shared within child view.doc, 1.0
- Big Product::Big Product\branch none floating\branch none floating 2::Big Product\Online Help::shared within child view.doc, 1.0
- Big Product::Big Product::Big Product\Source Code:;shared within child view.doc, 1.0
- Big Product::Big Product\branch none floating::Big Product\Source Code::shared within child view.doc, 1.0
- Big Product::Big Product\branch none floating\branch none floating 2::Big Product\Source Code::shared within child view,1.0

The next example shows that the file named `shareall.doc` existed only in the *branch all floating* view before it was shared to another view. The reference to the root folder starts the references that occurred as a result of the share operation. However, the recipient view could have been any of the other views, because the file would float up to the root and back down. On the way down, a second reference was created in the *branch all floating* view.

- Big Product::Big Product::branch all floating::Big Product\Marketing Documents::shareall.doc, 1.0
- Big Product::Big Product::Big Product::shareall.doc, 1.0

- `Big Product::Big Product\branch all floating::Big Product::shareall.doc, 1.0`
- `Big Product::Big Product\branch none floating::Big Product::shareall.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating2::Big Product::shareall.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating 2\branch none floating3::Big Product::shareall.doc, 1.0`

References Created by Moving Objects

When you move a folder or item from one location to another within the same view, StarTeam deletes the object at the old location and reinstates it at the new location. However, there can be side effects in that view's parents and children if any of the views are floating. This is because the copy at the old location is not deleted except in the current view. The parent and child views may end up with two references (one to the old location and one to the new location) instead of one to the new location.

Reference Hierarchy Examples

Suppose you move the file named `timeout.doc` from the `Marketing Documentation` folder to the `Timeout` folder in a given view that has no branching child views.

The following two examples show the references for this file before and after the move. The number of references is the same; only the path to the file has changed. The file has been deleted from its original location and added to its new location.

Before the move: `Big Product::Big Product::Big Product\Marketing Documents::timeout.doc, 1.0`

After the move: `Big Product::Big Product::Big Product\Source Code\Timeout::timeout.doc, 1.0`

However, suppose this view has a child view that was created without cutting off the connection to the parent (in other words the child view is branched and floating). In the child view, if the moved file has not yet branched, it is not deleted from its old location because you might really still want it here. However, it is added to the new location because it is perceived as a change to the parent that should be reflected in the child.

Notice that the file has only one reference in the parent but that it has two in the child view.

- `Big Product::Big Product::Big Product\Source Code\Timeout::timeout.doc, 1.0`
- `Big Product::Big Product\branched floating::Big Product\Marketing Documents::timeout.doc, 1.0`
- `Big Product::Big Product\branched floating::Big Product\Source Code\Timeout::timeout.doc, 1.0`

Some users sort items using folders. For example, they decide to create a series of folders in a view to classify change requests by criteria such as:

- Will definitely make the next release
- Are under consideration for the next release (time permitting)

These change requests are usually moved from the root folder to one of the sorting folders, or later rearranged and moved from one sorting folder to another. This is a convenience in the current view, but it can cause multiple references in a parent or child view. If the view hierarchy is deep, the current view's parents, grandparents, children, grandchildren, and so on may be affected. Users who use such systems usually create child views that do not float.

Floating Up and Down in the View Hierarchy

If the view hierarchy is deep (the root view has grandchildren, great-grandchildren and so on), the use of branching, floating views can cause a great deal of confusion. Suppose all the views except the root view branch and float. At its new location, the folder or item you move can float:

- Up the view hierarchy from the recipient view to the root view
- Down to all the children, grandchildren, and so on of the recipient view
- From the parents, grandparents, and so on of the recipient view to all of their other children

A move operation results in one fewer reference to the moved folder or item in the view from which it was moved, and one more reference to it in the recipient view.

The following example shows that a file named *move within parent view* was moved from one location in the root view to another location in that same view (which is why there is only one reference to it in that view). Originally, the file was referenced in five views. The move caused a new reference in all the child views of the root folder, giving each of them two references to the moved file (one reference in its original location and one in its new location).

- `Big Product::Big Product::Big Product\Source Code::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch all floating::Big Product::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating::Big Product::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating2::Big Product::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating2\branch none floating3::Big Product::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch all floating::Big Product\Source Code::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating::Big Product\Source Code::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating2::Big Product\Source Code::moved within parent view.doc, 1.0`
- `Big Product::Big Product\branch none floating\branch none floating2\branch none floating3::Big Product\Source Code::moved within parent view.doc, 1.0`

Folders and Items

The topics in this section provide information about how StarTeam manages folders and items and how to use them.

Overview of Folders and Paths

In StarTeam, three types of folders play important, yet dissimilar, roles.

Original workstation folder

Users set up this folder and its contents on a workstation, then use the **New Project Wizard** to create a new project. This folder, which may contain files and other folders, becomes the root folder of the new project – that is, it becomes the root folder of the project's initial or root view. StarTeam creates the project, the root view of the project, and root folder at the same time. The project, view, and root folder initially have the same name, although the name can be changed later

StarTeam folder

StarTeam uses the folders to group items in a project view. For example, a folder named `Source Code` can group source code files, requested changes to those files, and other related items. These folders can be created automatically at the same time as a project or added later by administrators or team members with the appropriate privileges. The hierarchy of folders in the current view appears as a folder tree in the project view window.

Working folder on the workstation

A working folder is actually a property of a StarTeam folder, but is quite different, as it is an object controlled by the operating system. It stores files that are copied or checked out from StarTeam or that will be added to StarTeam . A folder is an object controlled from within StarTeam. Data about it is stored in the database that holds all project data.

A project, its root view, and its root folder all have the same working folder. If additional views of the project are created, each view and its root folder have the same working folder. The working folder for the root folder always has an absolute path, which starts with the drive letter and lists the appropriate directories until it reaches the working folder itself.

You can add to a project at any time after it has been created. These folders can be moved from or shared with another StarTeam view or added from a workstation.

Understanding Working Folders

Understanding the relationship between application folders and their working folders is important because the working folder stores the files that you check in and check out.

Each folder has a default working folder from which you modify working files. For team members that use the same folders, the working folder structure on one person’s workstation is often the same as those on another person’s workstation.

When you check out a file, the application copies the requested file revision to the appropriate working folder. If the working folder does not already exist on your workstation, the application automatically creates it for you as you check out files that go in that folder.

The application expects you to add and check in new file revisions from those working folders. If the working folder does not exist on your workstation, you can create it manually or automatically using the Create Working Folders command. After the working folder exists, you can add files to it.

The exact location of a working folder is displayed as one of the application folder’s properties.

Alternate Working Folders

The view’s working folder may not be the optimal choice for all users. You, or any other user with the access rights to do so, can select a more useful location for the view’s working folder on your own workstation by designating an alternate working folder. For example, you might want to use a shorter path or a different drive letter. Remember that a working folder must point to a physically discrete location, such as a drive on your workstation or a personal directory on a shared file server. We do not recommend putting your settings on a mapped network drive.

The alternate working folder path for the view is specific to the workstation and user. For example, if you log onto the project as another user or use another workstation, your alternate working folder setting is not known.

When you designate an alternate working folder for the view, the path to the working folder for each child folder in the view may be similarly modified for your workstation.

For every folder in the hierarchy whose working folder is relative to the path of the view's working folder (as opposed to having an absolute path or an alternate working folder path of its own), your alternate path for the view's working folder becomes part of the paths to its child folders' working folders.

Folder Paths

StarTeam often stores working folder paths from development environment applications as relative paths. For example, `..\sc` may be the working folder for a project's `Source Code` folder. If you move a folder to another location in the hierarchy, its working folder may end up in an unexpected location. This result occurs because the application applies the relative path to the working folder path for the new parent folder. Therefore, if you move a folder, you may want to specify a working folder path that is not relative, to avoid accidentally changing the working folder path on users' workstations.

Folder Tab

StarTeam includes a component tab for folders called **Folder**. When selected, this tab displays a main menu item and context -menu that contains many of the same menu commands that you would use when working with files, change requests, requirements, and so on. It is possible to perform some operations on multiple selected **Folder** items, such as adding files to a view.

Folder states are represented by the following folder icons:

- Regular folder.
- Invisible Folder: Indicates a folder where the **Visible** property has been unchecked in the **Folder Properties** dialog box.
- Not-in-View Folder: Indicates a folder on your local disk that does not map to a folder.
- Missing Working Folder: Indicates that local working folders do not exist.
- Folder uses an alternate working folder path than the default one set up by the project.

Folders

The project or server administrator usually creates projects and project views. If you are a typical user, you routinely open a particular project view and manage *your* folders and their contents, such as files and change requests. Managing application folders is very similar to managing a project. You can create folders, delete folders, and modify their properties—if you have the correct access rights.

Folder Hierarchy

When you create projects, you typically select locations on your workstations as the working folders for those projects. The working folder designated for a project also becomes the working folder for the project's root view and for the root folder in that view's folder hierarchy.

StarTeam treats folders as both containers and items. You can group items within a project view by placing them into folders. For example, a folder named *Source Code* can contain source code files and requested changes to those files. You can create folders automatically when you create a project, or add folders after you create the project. Project or server administrators (or team leads – this all depends on your organization) usually create projects, but anyone can create projects if they have the correct access rights. See your server administrator if you have questions regarding the access rights assigned to you.

When you create a project, StarTeam automatically creates the parent or root folder for that project at the same time. It is actually the root folder of the project's root (or initial) view. The project, view, and this root folder initially have the same name (although those names can be changed).

Usually, the user who creates a project sets up a hierarchy of folders on a workstation before creating the project. The user designates the root folder of that hierarchy as the project's working folder. Then the application can automatically create an application folder for each of the child folders in the hierarchy. The child folder becomes the application folder's working folder.

If child application folders are created at the time the project is created, then:

- The application folders' working folders were part of an existing hierarchy on the project creator's workstation.
- Their names are the same as the names of their working folders, but they can be changed later.
- Their working folders remain hierarchically connected to the root folder's working folder. That is, if you change the path to the root folder's working folder, you also change the path to this folder (unless you manually set an absolute path for these working folders). In other words, the application stores a relative path to each child folder.

One of the most important properties to notice about your folder is its working folder. You will need to know where on your workstation the application will copy file revisions that you check out so that you locate those revisions as needed for modifications. A number of other operations can be performed on folders, such as moving a folder or changing its branching behavior.

A working folder is a property of the folder and represents the actual location on your workstation where StarTeam saves files that you check-out. Despite the fact that these are both called folders, the working folder and the folder are not identical. Their differentiating characteristics include:

- The path to the working folder can be totally different from the path within the application to the application folder.
- An application folder is an object controlled from within the application. The data associated with this folder is stored in the database that stores all the project data.
- A working folder is an object controlled by your operating system. It stores files that are checked out from the application.

A project, its root view, and the root folder of the root view all have the same working folder. For additional views, each view and its root folder have the same working folder.

The working folder for the view/root folder always has an absolute path (starting with the drive letter and specifically naming the folders at subsequent levels until you reach the working folder itself).

If you look at the properties for the root folder, you will see that the working folder is the same. However, it is displayed in the **Complete Working Folder Path** display box instead of the **Default** field. Since you can only change the working folder at the view level, all of the fields for the root folder's working folder are always disabled.

For the child folders that were created at the same time as the project, the application stores the path to each working folder as a relative path.

Understanding Default and Alternate Working Folders

Make sure that everyone is logged off from the StarTeam Server and that the StarTeam Server is locked before you change the Default Working Folder. It is just as critical to perform these actions as it is when you change custom fields or do anything else that affects all users.

When a view is created, a default location is specified for its working folder. If you change the Default Working Folder, not only the path to the working folder but the path to each child folder in the view may be similarly modified – not just for you, but for everyone working with that view. Therefore, before making such changes, it is important to understand the relationship of the working folder to the StarTeam view.

Default Working Folder path	Used by everyone sharing that view, unless they have specified an Alternate Working Folder path. Only change the Default Working Folder if you want to change the path for everyone who shares the view.
Alternate Working Folder path	Lets you specify a different location for your own working folder than the Default Working Folder. If you do not want to use the Default Working Folder path, specify an Alternate Working Folder path. Do not change the Default Working Folder. If you specify an Alternate Working Folder path, it is used instead of the Default Working Folder path. The Default Working Folder path must point to a location that is physically discrete for each

user, such as a drive on that user's workstation or a personal directory on a shared file server.

The working folder for the view's root folder has an absolute path (for example `C:\New Product`). The path used for the working folder of a child folder depends upon how the child folder was created and what changes have been made to the path since that time. Generally, the working folder for a child folder is relative to that of the view (that is, relative to the working folder used for the root folder). For example, suppose that the path to the view's working folder is `C:\New Product` and that the root folder has a child folder named `Online Help`. In this case, the path to the `Online Help` working folder would be"`C:\New Product\Online Help`. When the path to the view's working folder changes, the path to the child's working folder changes automatically.

If a new child folder is added to the view after it is created, the path to the child's working folder will usually be relative. However, if its working folder is on a different drive than the working folder for the root, its path will be absolute.

Granting Folder-Level Access Rights

Setting access rights at the folder level is usually done when you want to allow certain groups (but not other groups) to access a particular branch of the folder hierarchy. For example, you may want only the **Writers** group to be able to access the branch that has `User Manual` as its root folder.

Setting access rights at the folder and the item levels has more consequences than setting rights at higher levels. When a child view is derived from a parent view, as all reference and most branching views are, it initially contains objects that belong to its parent. In branching views, these objects can branch into new objects that exist only in the child view. Just as a new view has no view-level access rights, folders and items that branch into new objects initially have no access rights at the folder or item level.

This Folder and Child Folder Nodes

The folder level has two nodes: This Folder, for the selected folder, and Child Folders, for the other folders in the folder hierarchy of the branch. This feature allows you to set different access rights for each node.

In the client, the root folder of a view can be indistinguishable from that view. If the view is the root (or initial) view of a project, the root folder can be indistinguishable from that project.

Using the **This Folder** node to set access rights for the root folder can therefore affect a user’s access to a view. If the view is the root view, it can also affect the user’s access to the project. Therefore, most administrators avoid setting folder-level access rights on a root folder, as these rights may interfere with view-level or project-level rights.

For example, suppose the **Developers** group is not granted the right to see the User Manual folder and that this folder is the root of a reference view. Then members of the **Developers** group cannot open that view, even if view-level access rights allow them to see the view. An error message appears when they try to open the view. Users who can see a project but not its root view also see an error message.

Access Rights of Child Views

If a child view includes child folders that have access rights in the parent view, its access rights depend upon whether it is a reference view or a branching view.

Access Rights in a Reference View

The access rights in a reference view at the folder level are not independent of the access rights at the folder level in the parent view, as no branching will ever occur. You can see these access rights from either view if you have the rights to do so.

If you change access rights in the reference view, you simultaneously change the access rights in the parent view (and vice versa) because the folder in the reference view is the same object as the folder in the parent view.

Access Rights in a Branching View

If the child view is a branching view, the access rights in the child view at the folder level are independent of the access rights at the folder level in the parent view, but only after the folder in the branching view actually branches.

Initially, any folder you see in the branching view is the same object that exists in the parent view. Therefore, it has the same access rights in both views. Initially, you can change access rights in the parent view (and vice versa), because the folder in the branching view is the same object as the folder in the parent view. Once the folder branches, however, a new object for that folder is created in the branching view. This object begins a life cycle of its own and no longer has any access rights at the folder level.

Note: Remember that branching a folder does not branch any of the folder's contents. Each item is the folder is treated separately.

The behavior of folders in a branching view affects the access rights:

- If a folder branches on change and you change one of its properties, its revision number changes. When the folder branches, it becomes a new object in the repository and no longer has any access rights at the folder level.
- If a folder does not branch on change and you change one of its properties, the revision number changes, but no new object is created. In this case, the folder retains its access rights in both views. Because both views still contain the same object, changes you make to the folder's access rights in one view also change that folder's access rights in the other view.

Folder Access Rights

When you select the **Folder Tree** > **Advanced** > **Access Rights** command to display the **Folder Access Rights** dialog box, you see two folder nodes. The rights available from **This Folder** node apply to the selected folder only. The rights available from the **Child Folders** node apply to all the child folders of the selected folder. The dialog and following table refer to the current folder. The table describes the access rights that are available from the **This Folder** node in the **Folder Access Rights** dialog box.

Note: Because **This Folder** has no **Generic item container** subcategory for access rights, container rights for **This Folder** are on its **Child Folders** node. If **This Folder** is the root folder, these rights are set on the **Child Folders** node of the **View Access Rights** dialog box.

Generic object rights

See item and its properties

View this folder's **Name**, **Exclude**, and **Files** tabs, which become available when **Folder** > **Properties** is selected. The **History** tab is controlled by the `See folder history` access right. The **Link** tab is controlled by the `See folder links` access right.

Modify properties

Change folder properties on the folder **Name** and **Exclude** tabs. Properties include folder name, description, use of inherited and local exclude lists, and contents of the local exclude list. If the folder is not a root folder, the working folder and alternate working folder settings are also properties. For root folders, the working folders are view properties and not controlled by this access right.

Delete from folder

Delete this folder from its parent folder. Be aware that if you can delete any of this folder's parent folders, you can still delete this folder.

Change item access rights

Change the access rights for this folder. If you change this setting, be sure that you remain one of the users who can change access rights.

See history See this folder’s **History** tab, which is available when **Folder** > **Properties** is selected.

Perform maintenance Change the revision comments for past revisions.

Set exclusive locks Lock folders exclusively.

Break exclusive locks Remove someone else’s exclusive lock on the folders.

Label Rights

Attach/Adjust view labels Add a view label to this folder. Move a view label from one revision of this folder to another. This right controls direct manipulation of labels for this folder at the folder level. It does not stop users from attaching a view label to this folder when a view label is created.

Detach view labels Remove a view label from this folder. Be aware that if users can delete view labels, they can detach a view label from this folder by deleting the view label from the view, regardless of the setting for this right.

Attach/Adjust revision labels Add a revision label to this folder. Move a revision label from one revision of this folder to another. This right controls direct manipulation of revision labels for this folder at the folder level.

Detach revision labels Remove a revision label from this folder. Be aware that if users can delete revision labels, they can detach a revision label from this folder by deleting the revision label from the view, regardless of the setting for this right.

Link Rights

See links See the links involving this folder.

Create links Link this folder to other folders and items.

Modify links Change a link for this folder.

Delete links Delete a link for this folder.

Child Folder Access Rights

When you select the **Child Folders** node from the **Folder Access Rights** dialog box, the available rights apply to the child folders of the selected folder. The **Child Folders** node is also available from the **View Access Rights** dialog box and the **Project Access Rights** dialog box. In these cases, the rights apply to all child folders in the current view or all the child folders in the project, respectively.

Below is a description of the access rights available from the **Child Folders** nodes in the **Project Access Rights**, **View Access Rights**, or **Folder Access Rights** dialog boxes.

Generic item rights

See item and its properties See the selected folder’s child folders or the selected project’s or view’s folders in the folder hierarchy in the left pane on the screen. You can also view the **Name** and **Exclude Properties** dialogs, which open when **Folder** > **Properties** is selected. The **History** tab is controlled by the **See folder history** access right.

Modify properties

Change folder properties on the **Name** and **Exclude** tabs for child folders. The properties include the folder’s name, description, use of inherited and local exclude lists, and the contents of the local exclude list. If a child folder is not a root folder, the working folder and alternate working folder settings are folder properties. If it is the root folder, the working folders are view properties and not controlled by this access right.

Delete from folder

Delete the selected folder’s child folders or the selected project’s or view’s folders from their parent folders. Be aware that if you can delete any of this folder’s parent folders, you can still delete this folder.

Change item access rights

Change the access rights for the selected folder’s child folders or the selected project’s or view’s folders. If you change this setting, be sure that you remain one of the users who can change access rights.

See history

See the **History** tab, which is available when **Folder** > **Properties** is selected. This action applies to the selected folder’s child folders or the selected project’s or view’s folders.

Perform maintenance

Change the revision comments for past revisions.

Set exclusive locks

Lock child folders exclusively.

Break exclusive locks

Remove someone else’s exclusive lock on the child folders.

Label rights

Attach/ Adjust view labels

Add a view label to the selected folder’s child folders or the selected project‘s or view’s folders. Move a view label from one revision of a child folder to another. This right controls direct manipulation of view labels for child folders at the folder level. It does not stop users from attaching a view label to child folders when a view label is created.

Detach view labels

Remove a view label from the selected folder’s child folders or the selected project’s or view’s folders. Be aware that if users can delete view labels, they can detach a view label from child folders by deleting the view label from the view, regardless of the setting of this right.

Attach/ Adjust revision labels

Add a revision label to the selected folder’s child folders or the selected project’s or view’s folders. Move a revision label from one revision of a child folder to another. This right controls direct manipulation of revision labels for child folders at the folder level.

Detach revision labels

Remove a revision label from the selected folder’s child folders or the selected project’s or view’s folders. Be aware that if users can delete revision labels, they can detach a revision label from this folder by deleting the revision label from the view, regardless of the setting of this right.

Link rights

See links

See the links involving the selected folder’s child folders or the selected project’s or view’s folders.

Create links

Link the selected folder’s child folders or the selected project’s or view’s folders to other folders and items.

Modify links Change a link for the selected folder’s child folders or the selected project‘s or view’s folders.

Delete links Delete a link for the selected folder’s child folders or the selected project’s or view’s folders.

Generic item container rights

Create and place in folder Create a folder in a parent folder, view, or project in which the **Child Folder Access Rights** dialog box has this option.

Share/Move out of folder Share or move a folder in a parent folder, view, or project if its Child Folder Access Rights dialog has this option. Be aware that the access rights set for that folder and its contents, along with any rights set for specific child folders and items within that branch of the folder hierarchy, accompany the folder into the new folder.

Change behavior or configuration Change the branching ability and configuration of folders that reside in a parent folder, view, or project if its **Child Folder Access Rights** dialog box has this option.

Folder Properties

This topic presents the folder properties and their descriptions as displayed in the **Folder Properties** dialog box. The **Folder Properties** dialog box contains the following tabbed pages of properties.

Name tab

The following properties are on the **Name** tab.

Name Displays the name of the file.

Description Displays the file description.

Created By Displays the name of the person who created the file.

(Created) On Displays the date on which the file was created.

Visible Indicates whether the file is to be visible in the view or not.

Working Folder Displays the path to the default and alternate working folders.

- **Default** This path location points to the default working folder, and applies to everyone accessing the project repository. DO NOT CHANGE this path unless you are a project administrator.
- **Alternate** This path points to an alternate working folder on your machine. Specifying an alternate working folder affects only you, not any of the other team members.

Complete Working Folder Path Displays the complete path to the selected working folder (default or alternate.)

Exclude tab

The following properties are on the **Exclude** tab.

Files To Be Excluded Indicates which files or types of files to exclude from visibility in the folder. The exclude list has no effect on files that are already part of the project. It only affects those with **Not In**

View as their status. Exclude lists can be inherited from parent folders. **Exclude** options include:

Inherit And Use Local Exclude List	Indicates that files matching the exclude list specifications set for this folder and those of its parent folder will be excluded.
Use Local Exclude List	Indicates that files matching the only specifications set for the exclude list of this folder will be excluded.
No Exclude List	Specifies that all files are included in the folder.

Local Exclude List Displays the exclude specifications to use for excluding files from this folder. The exclude list is limited to a maximum of 255 characters. It contains file specifications (using the standard * and ? wild cards), separated by commas, spaces, or semicolons. To include a comma, space, or semicolon as part of the specification, enclose the specification in double-quotes. For example,

```
*.exe,*.dll p*z.doc;*.t?t "test *.*"
```

Inherited Exclude List Displays the exclude specifications to use from the parent folder for excluding files from this folder.

History tab

The **History** tab displays all the revisions of the folder. The following properties are displayed on the **History** page for each revision.

View Displays the name of the view to which this folder belongs.

Revision Displays the file revision number.

Modified By Displays the name of the person who created the folder.

Modified Time Displays the date and time the revision created.

Comment Displays a comment explaining why the revision was created.

Dot Notation Displays the branch number of the revision.

Link tab

The following properties are on the **Link** tab which displays all the links to this folder.

Created By Displays the name of the person who created the link to the folder.

Created On Displays the date on which the link was created.

View Displays the name of the current view if the link was created in the current view, or displays the name of view where the link was created and from which the link is shared.

Folder Displays the name of the folder in which the folder or item in the link resides.

Item Type Identifies the type of item to which the target end of the link is attached. This item is listed in the link list.

Item Identifies the item to which the target end of the link is attached. It is identified by its folder name, file name, change request number, task number, topic number, or requirement number.

Item Details Describes the item, using a folder description, file description, change request synopsis, task name, topic title, or requirement name.

Item Version Displays the version number of the target end of the link if that revision is in the current view. When no revision number is displayed in the column, that end of the link is floating rather than pinned.

Selection Version Displays the version number of the source end of the link if that revision is in the current view. When no revision number is displayed in the column, that end of the link is floating rather than pinned.

Comment Displays a comment about this particular link.

File Status Displays the status of a file that is linked to the folder.

Locked By Displays the name of the person who locked the file linked to the folder.

Folder Path Shows folder path information only when the linked item is in the same view. Otherwise, it displays the message, "`Unavailable. Item in another view.`"

Folder Fields

This section lists all the folder fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or StarTeam Notification Agent. Reports can use any field name.

Branch On Change (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `BranchOnChange`.

Indicates whether the item will branch when it changes.

The value is `No` if the item's behavior is not set to **Branch On Change**. Reasons for this may be:

- The item is in the root or a reference view and the **Branch On Change** feature is disabled.
- The item is in a branching view but has already branched as a result of a change, which, in turn, results in the **Branch On Change** feature becoming disabled.
- The item is in a branching view, but its behavior currently does not permit it to branch on change. This means that modifications are checked into the parent view.

Note: If the value is `No`, the value of the **Branch State** explains the `No`.

Branch State (Advanced)

Values: `Branched`, `Not Branched`, `Root`.

Internal Identifier: `BranchState`.

Indicates whether an item has branched in the child view, is still unbranched (and therefore is part of the parent view), or was created in the view in which it resides.

The values `Branched` and `Not Branched` apply to items in branching views. The value `Root` applies to items created in the view in which the item currently resides.

If the view is a reference view, it reflects the state of the item in the reference view’s parent.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those

2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created, or the time at which the label associated with the promotion state was created.

Created By

Values: list of users, <`None`>.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Creating Project

Values:

Internal Identifier: `CreatingProject`

Deleted By

Values: list of users, <`None`>.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Description

Values: `text`.

Internal Identifier: `Description`.

The description provided for an item at the time it was added to the view, including any later edits to it.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

Dot Notation ID (Advanced)

Values: `number`.

Internal Identifier: `DotNotationID`.

The ID assigned to a particular branch revision number. For example, if a folder was added to the current view (as opposed to inherited by the current view), its branch revision number is `1.x` and its branch revision ID is `0`. If a folder was branched in the current view, its branch revision ID is dependent on the revision number in the parent view and the number of IDs already assigned in the current view. For example, if a folder's revision number in the parent view is `1.7` at the time of the branch, and another folder with that same parent revision number was given the Branch Revision ID `6`, this folder will also be given the Branch Revision ID `6`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

Exclude Flags

Values: `Inherit and Local Exclude List`, `Local Exclude List`, `No Exclude List`.

Internal Identifier: `ExcludeFlags`.

The flag which specifies the types of filse to be excluded from the folder.

Exclude Spec

Values: `text`.

Internal Identifier: `ExcludeSpec`.

The file specification (using the standard `*` and `?` wild cards), separated by commas, spaces or semicolons. To include a comma, space, or semicolon as part of the specification, enclose the specification in double quotes.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

Is Action Overidden?

Values: `No`, `Yes`.

Internal Identifier: `Is Action Overidden?` (contains spaces).

Local Path

Values: `text`.

Internal Identifier: `Local Path` (contains spaces).

The local path to the folder. This is the path to the working folder.

Locked By

Values: list of users, <`None`>.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

Modified By

Values: list of users, <`None`>.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

Name

Values: `text`.

Internal Identifier: `Name`.

Displays the name of the item.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Parent Branch Revision (Advanced)

Values: `number`.

Internal Identifier: `ParentRevision`.

The last digit in the branch revision number before an item branched. For example, if this number is `7`, the branch revision was `1.7` at the time the item branched (becoming `1.7.1.0`, as seen in the item's history). This number is `-1` if an item was not inherited from the parent view.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Parent Revision (Advanced)

Values: `number`.

Internal Identifier: `PathRevision`.

The revision number at which an item branched. For example, if this number is `8`, this item's revision number in the parent view was `8` at the time the item branched. The history should show that revision `9` in the first revision in the current view. This number is `0` if this item was not inherited from the parent view.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item's configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Root Object ID (Advanced)

Values: `number`.

Internal Identifier: `RootObjectID`.

The object ID of the oldest ancestor of an item. For example, if an item was not inherited from a parent view, the root object ID is the same as its object ID. If it was inherited from a parent view, the root object ID is the Parent ID, or the item’s Parent ID.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item’s properties or contents. Additional text is stored in the **Comment** field.

Status

Values: `Current`, `Deleted on Disk`, `Deleted on Server`, `Merge`, `Missing`, `Modified`, `Not In View`, `Out Of Date`, `Unknown`.

Internal Identifier: `Status`.

Indicates the relationship between the copy of an item in your working folder and the tip revision in the repository.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is `2`.

View

Values: list of views, `<None>`.

Internal Identifier: `ViewID`.

The name of the view in which the item last branched. For example, if an item is inherited from a parent view but is branched in a child view, the value of this field in the child view changes from the name of the parent view to the name of the child view for the revision that branched and subsequent revisions in the child view.

*****Working Folder**

Values: `number`.

Internal Identifier: `WorkingFolder`.

The name of the working folder.

Adding Folders to Views

Use the following procedures to add folders to views.

Adding a New Folder to a View

1. Right-click the root folder of the view and choose **New**. The **New Folder Wizard** opens.
2. In the **New Folder Wizard**, select a parent folder for the new folder in the folder tree and click **Next**. The new folder will be created as a child of the selected folder, and the **Folder Name** page of the wizard displays.
3. Type a name of up to 254 characters for the child folder.
4. Do one of the following:

    - Leave the **Working Folder** text box blank. The application creates the working folder using the name of the new folder and the path to its parent's working folder.
    - Type in or browse to the path for an existing working folder in the **Working Folder** text box. When you browse for a path, you create an absolute path to this folder's working folder.

      Note: If you want this folder's working folder to be relative to its parent folder's working folder, type the addition to that path in this text box. Do not browse.
5. Type a description of up to 254 characters for the new folder in the **Folder Description** text box, and click **Next**. The **Child Folders** page of the **New Folder Wizard** displays the new folder in the **New Folder's Child Folders** box. If the working folder has child folders, an application folder is created for each of them.
6. Do one of the following:

    - To exclude a child folder from your project, select the folder and click **Exclude**.
    - To exclude all child folders, click **Exclude All**.
    - To re-display folders you have excluded, click **Reset Folders**.
7. Click **Next**. The **Folders** page of the **New Folder Wizard** displays the view's folder hierarchy with the new child folder.
8. Click **Finish**.

Adding Not-in-View Folders to a Project

1. Choose **Folder Tree** > **Show Not-in-View Folders** to make Not-In-View folders visible in the client. With this item checked on the **Folder Tree** menu, you can see Not-in-View folders in the folder hierarchy on the left, and on the **Folder** tab in the upper pane.
2. Select the Not-in-View folder(s) you want add to your view.

   Note: You can select multiple sub-folders on the **Folder** tab using **Ctrl+click** , or **Shift+Click +Arrow** .
3. Right-click the selected files and choose **Add to View**.
4. Select the files in the folder you just added on the **File** tab and choose **Add Files**.

Note: This operation allows you to add the selected folder (and any Not-In-View parent folders) to the project folder tree. This is an alternate way to create folders instead of using the **New Folder** wizard. The folders it creates have the same folder name and working folder path.

Setting Folder Component Access Rights

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Right-click the folder and choose **Advanced** > **Access Rights** .

Note: You must have the administrative rights to change component access rights for a project.

3. In the **Folder Access Rights** dialog box, select a component from the **Category** list, then click **Add**.
4. In the **Assign Access Rights To** dialog box, select a group/user and **Grant** or **Deny** them access.
5. Click **OK** to return to the **Folder Access Rights** dialog box.
6. Select the group or user you just added to the **Users and Groups** list, then check the desired options in the **Rights** column, or click **Select All**.
7. Choose **Grant** or **Deny** to specify what action to take for the selected users and rights.
8. Continue this process until you have finished granting or denying access rights to the appropriate users and groups, then click **OK**.

Creating a Working Folder

In StarTeam, each of the child folders in the view has its own working folder, which is generally relative to the path of the root working folder. When you check out a file, StarTeam copies the requested file revision to the appropriate working folder. If the working folder does not currently exist on your workstation, StarTeam automatically creates it for you when you check out files.

You can also add new files to StarTeam from a working folder. If the appropriate folder does not yet exist on your workstation, you can create it automatically by using the **Create Working Folders** command. Once the working folder exists, you can place files in it and add them to StarTeam.

Important: The default working folder must point to a location that is physically discrete for each user, such as a drive on that user's workstation or a personal directory on a shared file server.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.

   Note: A folder that exists in the StarTeam repository, but not on your local workstation, is represented by a folder icon with an exclamation mark on it.
2. Do one of the following:
   - Check out a file from the folder. StarTeam will automatically create a working folder with the same name and the same path.
   - Right-click the folder and choose **Create Working Folders**. After the working folder exists, you can copy files to it or create files in it and add them to the StarTeam repository.

Note: If the working folder path for a shared or moved folder exceeds the operating system's maximum working folder path length of 254 characters (including [\ ] backslashes), StarTeam will not allow you to create the working folder and displays an error message.

Opening a Local Folder

The following procedures describe how to open a local folder in a file browser so you can perform basic file and folder management tasks.

Opening a Local Folder from a Folder Selection

1. In StarTeam, select from one to five folders you want to open. You can select one folder in the folder hierarchy tree or up to five folders on the **Folder** tab in the upper pane.
2. Right-click the selected folder and choose **Open Local Folder**.

This opens a **Windows Explorer** for each location on disk that corresponds to a selected folder. This applies to all folders except those whose status is **Missing** since their local folders do not exist.

Opening a Local Folder from a File Selected in StarTeam

1. On the **File** tab in the upper pane, select one to five files.
2. Choose **File** > **Open Containing Folder** .

This opens a **Windows Explorer** to each folder in which a selected file exists. This applies to all folders except those whose status is **Missing** since their local folders do not exist.

Changing a View's Default and Alternate Working Folders

Make sure that everyone is logged off from the server and that the server is locked before you change the *Default Working Folder*. It is just as critical to perform these actions as it is when you change custom fields or do anything else that affects all users.

When you change the **Default Working Folder**, not only the path to the working folder but the path to each child folder in the view may be similarly modified—not just for you, but for everyone working with that view.

Caution: Do not change the **Default Working Folder** unless you are a project administrator. These default settings affect all users and incorrect settings cause other users to be unable to check out StarTeam files. The default settings should only be set to the name of the folder. If you want to use a different location for your working folder than the **Default Working Folder** path, specify an **Alternate Working Folder** path.

1. Choose **View** > **Properties** to open the **View Properties** dialog box.
2. Click the **Name** tab.
3. Do one of the following:
   - Select **Alternate** to create a different working folder for only yourself.
   - If you are a project administrator, select **Default** to specify the default repository path for all users.
4. Type the name of a new working folder or browse for a path to a working folder. If you browse for the path, it becomes an absolute path. This path can be edited, however, to enable you to work on a computer that uses a different letter for its hard drive.

   Note: It is important that the **Default Working Folder** point to a location that is physically discrete for each user, such as a drive on that user's workstation or a personal directory on a shared file server.

Changing Name or Description of Folders and Items

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Right-click the selected item and choose **Properties**.
3. Change the **Name**/**Description** for the folder or item and click **OK**. The folder or item name/description is changed in both the StarTeam repository and in your working folder.

Displaying Historical Properties

You can display the properties for a previous item revision using the **History** pane.

1. Select the item in the upper pane whose historical properties you want to view.

   You can select an item on any of the component tabs in the upper pane except the **Audit** tab.
2. Click the **History** tab in the lower pane.
3. Right-click the historical revision in the lower pane and choose **Properties**.

The item's **Properties** dialog box displays the properties of that revision.

Comparing Properties

Using the **History** pane, you can compare item properties of the tip revision with a previous revision.

1. Select an item in the upper pane.

   You can select an item on any of the component tabs in the upper pane except the **Audit** tab.
2. Click the **History** tab and select any revision except the latest (tip) revision.
3. Right-click the selection and choose **Compare** > **Properties** . The **Compare Properties** window displays the properties of the revisions in adjacent columns.

Comparing Historical Properties

You can compare the properties of two historical revisions of an item using the **History** pane.

1. Select an item in the upper pane.

   You can select an item on any of the component tabs in the upper pane except the **Audit** tab.
2. Click the **History** tab and select the two item revisions.

   Use **Ctrl+Click** to select the second revision.
3. Right-click the selection and choose **Compare** > **Properties** .

The **Compare Properties** window displays the properties of the revisions in adjacent columns.

Note: You can also select two non-file items, such as a change request, or two revisions of the same non-file item, and choose **Tools** > **Compare** . This opens the embedded double-paned compare view at the bottom of the window and displays the properties of the two selected items.

Comparing Properties of Two Non-file Items

You can compare the properties of two non-file items or different revisions of the same item.

1. Select the two non-file type items whose properties you want to compare, such as two change requests in the **Upper** pane, or select two revisions of the same change request in the **History** pane.
2. Choose **Tools** > **Compare** from the main menu. This opens the embedded compare panes at the bottom of the windows, and displays the properties of the two selected items. You can only view the properties here. You cannot change the properties.

Configuring (Rolling Back) Folders and Items

You can configure (or roll back) an individual folder or item to a specific view label, promotion state, or date and time. Essentially, all rollbacks are made to a particular date and time. For example, if you roll back to a view label, you essentially roll back to the revision of the folder or item that existed on the date and time at which that label was attached. Unlike a view, a folder or item retains its roll-back configuration until you manually change it or until the folder or item branches. When you close the view, the folder or item does not immediately return to its current configuration.

Rolling back a folder does not re-configure any of the items or child folders associated with it. It only rolls back the folder properties to the values they had at the configuration time. Depending on the folder behavior, the folder may become read-only, in which case its properties cannot be changed.

The configuration of a folder affects the new items or child folders that can float into it. For example, in a floating branch view, you can keep items from floating into a particular folder by configuring the folder to a particular label, promotion state, or point in the past. Later, you can re-configure the folder to floating so that it can receive new items from its parent. However, the items added to the parent while the folder was not floating will never automatically go into the folder. They must be manually shared. To freeze a folder or item at a certain point in time so that it cannot be changed:

- Change its configuration to a point in the past.
- Make sure that its branching behavior is either disabled or not set to **Branch on Change**.

Caution: There is no way to locate folders that have been configured to a point in the past unless you make a note of them. Use this feature with caution.

Rolling Back a Folder or Item

1. Do one of the following:
   - Select a folder from the folder tree or in the upper pane on the **Folder** tab.
   - Select one or more items in the upper pane.
2. Right-click the selected items and choose **Advanced** > **Behavior** . The **Item Behavior** dialog box opens.
3. On the **Configuration** tab, select a configuration option:

   Labeled Revision This option uses the folder or item revision with the specified view label as the tip revision. Existing view labels are listed in reverse chronological order based on the time for which they were created. This option is disabled if the view has no labels defined.

   Promotion State Configuration This option uses the folder or item revision with the view label assigned to the selected promotion state as the tip revision. This option is disabled if this view has no promotion states defined.

   Revision As Of This option uses the folder or item revision just prior to the specified point in time as the tip revision. It defaults to the current date and time, but you can select a date and time in the past, as long as it is after the time when the folder or item was created.
4. Click **OK**.

Returning to the Current Folder or Item Configuration

1. Do one of the following:
   - Select a folder from the folder tree or in the upper pane on the **Folder** tab.
   - Select one or more items in the upper pane.
2. Right-click the selected items and choose **Advanced** > **Behavior** . The **Item Behavior** dialog box opens.
3. On the **Configuration** tab, select **Floating**, then click **OK**.

Identifying Rolled Back Items

1. Right-click the column headers in the upper pane and choose **Show Fields**.
2. Select the **Configuration Time** field in the **Available Fields** list and click **Add**. This displays the **Configuration Time** column the upper pane.
3. Sort based on the **Configuration Time** column, because only items with configuration times in the past have any values in this field. If the **Configuration Time** matches the time for a view label, the item may be configured to that view label or to a promotion state based on that view label.

   Other fields that may be of interest are:

Branch On Change Uses a **Yes** or **No** value to indicate whether a file will branch when it changes. If **Branch On Change** is **No** and **Configuration Time** has a value, the item is frozen (read-only).

Branch State Indicates whether an item has branched in the child view, is still unbranched and part of the parent view, or is in the root view for the project and, therefore cannot be branched. Its values are **Branched**, **Not Branched**, and **Root**.

Deleting Folders and Items

The procedures below explain how to delete folders and items in the StarTeam client. They also explain how to delete the your working folders without deleting the folders in the view or project.

Deleting a StarTeam Folder

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Right-click it and choose **Delete**.
3. A dialog box opens prompting you to confirm the deletion.
4. Optionally, check **Delete Working Folders** to include your local folder and its contents.
5. Click **Yes**. StarTeam deletes the selected folder and all its files and sub-folders.

Deleting a Local Folder

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Right-click the selected folder and choose **Delete Local Folders**. A dialog box opens prompting you to confirm the deletion.
3. Click **Yes**.

Deleting an Item

1. Click a component tab in the upper pane, such as **File**, **Change Request**, **Requirement**, **Topic**, or **Task**, and select one or more items.
2. Right-click it and choose **Delete**.

Displaying Item Details

You can view an item's details quickly using the **Detail** tab in the lower pane. The **Detail** pane displays the item properties and their values in a two-column list.

The properties that display in the **Detail** pane for file, change request, child folder, and audit items are determined by which fields you choose to display in the upper pane/which filters you use for the component tab.

1. Click a component tab.
2. Select an item in the upper pane.
3. Click the **Detail** tab in the lower pane.

Displaying Location References

Because of manual sharing and because views are children of other views, a folder or item can be associated with more than one project, view, or parent folder (within the same server configuration). Each instance of the folder or item has a reference to its tip revision.

To view folder references, you open a separate dialog; to view item references, you use the project view window and the **Reference** tab on the lower pane.

You can view references for any of the following:

- Folders and past revisions of a folder.
- Items and past revisions of an item.

Viewing Folder References

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Right-click the selected folder and choose **Advanced** > **References** . The **Folder References** dialog box opens and displays a tree that indicates which project views reference this folder, and their relationship to each other.

Viewing References for Past Revisions of a Folder

1. Right-click the folder in the folder hierarchy tree and choose **Properties**.
2. Click the **History** tab in the lower pane.

   Note: There is no **History** tab if you do not have the access rights that allow you to see the folder history.
3. Select the revision in the **History** list.
4. Right-click the selected revision and choose **References**.
5. The **References** dialog box appears.

Viewing Item References

1. Select an item in the upper pane.
2. Click the **Reference** tab beneath the lower pane.
3. Data similar to that in the **Folder References** displays in the lower pane. The **Reference** pane has no context menu.

Viewing References for Past Revisions of an Item

1. Select an item in the upper pane.
2. Click the **History** tab in the lower pane.
3. Select the revision in the **History** list.
4. Right-click the selected revision and choose **References**.
5. The **References** dialog box appears.

Emailing Item Properties

You can send a text representation of selected items (except files) as an email message, along with additional text. The information sent for each item includes the fields displayed in the upper pane. For items such as change requests, the item's properties, which are the same as its contents, are sent in the email. For files, only the properties can be sent. However, a shortcut to the item can be included.

Items are considered to have been sent by the application, not by you. Therefore, you may want to copy yourself on the email. Otherwise, you will not receive the message.

Note: If you set up a filter and email an item, only the fields displayed by the filter are sent to the recipient.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Select an item in the upper pane.

4. Do one of the following:
   - Click the **Send** button on the toolbar. (If you have selected a file component, this button does not appear.)
   - Right-click the selected item and choose **Send To**.

   The **Send To** dialog box opens.
5. Click **To** or **CC** to open a dialog box for selecting the primary or secondary email recipients.

   Select the email recipients by moving the team member names from the **Available Users** to **Selected Users** list and click **OK**.
6. Type a **Subject**.
7. Optionally, check **Send A Copy To Myself** if you want to receive a copy of the email.
8. Optionally, check **Attach Item Shortcut** to include a shortcut to this specific item in the email.
9. Type any additional information in the **Add Text To The Mail Message** text box.
10. Click **Send Now** to send the message.

Note: Unlike automatic email notification, this message will not display the word “notification” in the subject line. Do not confuse email messages sent by individuals with email notification messages automatically sent by the StarTeam Server. If your administrator has enabled email notification, you will automatically receive email messages notifying you about items for which you are responsible and topics for which you are listed as a recipient.

Finding Items

You can search all items displayed in the upper pane for the data contained in any displayed field. For example, you can locate a change request by its number or search for a file with a particular name, status, time stamp, or size.

If you want to search a field that is not displayed, double-click a column header in the upper pane, select **Show Fields** from the context menu, select the desired field, and click **Add**.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Enter `CTRL+F`. The **Find** dialog box opens.
4. Type part or all of the data in the **Search For** field.

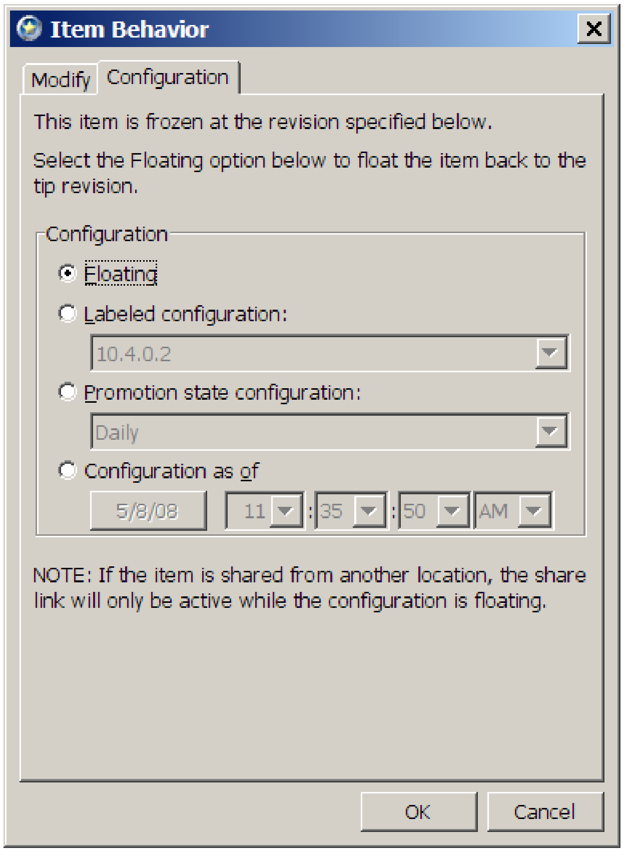
   Note: Wild cards are not supported.
5. Select **Forward** to search the upper pane from the top to the bottom, or select **Backward** to search the upper pane from the bottom to the top.
6. Select **Starting at: Currently Selected Item** to begin searching from the item that is presently selected, or select **Starting at: First Item** to search from the first item in the upper pane.
7. Select either **All Displayed Fields** or **This Field**. If you select **This field**, select the field for which you want to search from the list.
8. Check **Match Case** if a case-sensitive search is appropriate.
9. Click **Find** to search.

Tip: Use **F3** to find the next item that matches the search text and **Shift+F3** to find the previous item that matches the search text.

Floating Items

Floating is a StarTeam feature that needs careful management. Floating is a configuration option available for all items on the **Configuration** tab of the **Item Behavior** dialog box.

The alternative is a pinned configuration. Items can be pinned to a label, promotion state, or a point of time in the past. The StarTeam field named **Configuration Time** shows you the configuration of an item. The field is blank when the item is floating. The field contains a date and time if the item is pinned. That date and time may be the time that the view was created, the time the item was shared by a VCM operation, a time selected by a user on the **Configuration** tab in the **Item Behavior** dialog box, the time of the promotion state to which the item was pinned, or the time of the label to which the item was pinned.



This item is frozen at the revision specified below.

Select the Floating option below to float the item back to the tip revision.

Configuration

● Floating

○ Labeled configuration: 10.4.0.2

○ Promotion state configuration: Daily

○ Configuration as of 5/8/08 11 : 35 : 50 AM

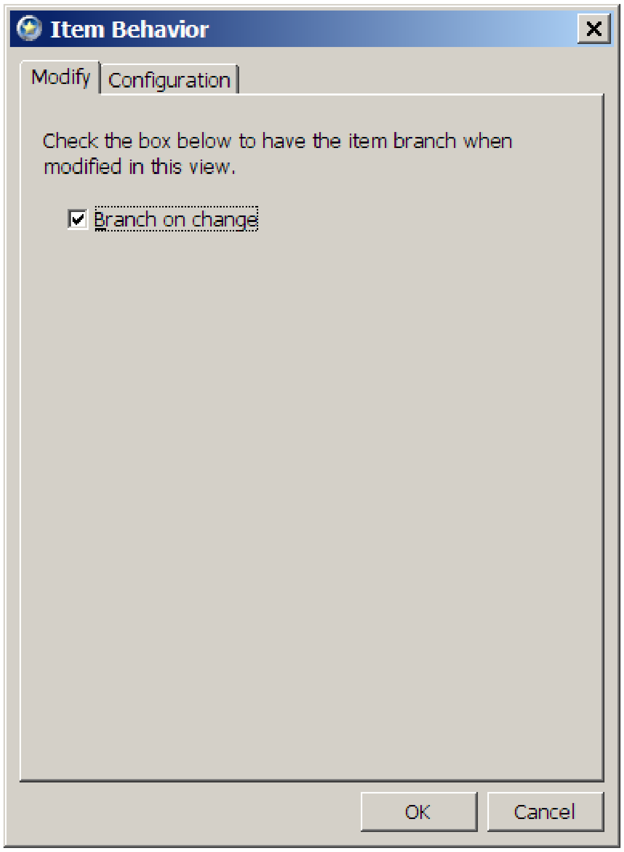
NOTE: If the item is shared from another location, the share link will only be active while the configuration is floating.

Effects on an Item Caused by a Floating Configuration

The effects on an item caused by a floating configuration depend on the item's behavior, for example if it is an item type that can branch: folder, file, or change request, and on the behavior of the folder in which it resides. The StarTeam field named **Branch State** shows you whether an item that can branch is a *Root*, *Branched*, or *Unbranched* item.

- A Root item is the root of a reference tree. It always has the dot notation 1.x.
- A Branched item is not the root reference in the reference tree and it has branched in the location where it resides. In its history, you should be able to see at least one place where its dot notation changed form 1.x to 1.x.y.0. Every time that an item branches two more digits are added to its dot notation. An item branches only once in each location, but it may have been shared to many locations. The reference tree shows those locations.
- An Unbranched item is not the root reference in the reference tree, and it has not yet branched in the location where it resides. The StarTeam field name **Branch On Change** displays a **Yes** or **No** for unbranched items indicating whether the item is set to branch when it is changed. There is a check box on the Behavior dialog named **Branch on change** which is selected when an item is set to branch on change.

The following dialog shows that an item is set to branch on change. Its behavior can be changed from this dialog, but "Branch On Change" = yes is the preferred behavior:



Check the box below to have the item branch when modified in this view.

☑ Branch on change

If this item’s **Branch State** had been `Root` or `Branched`, the check box would be disabled.

If this item’s **Branch State** had been `Root` or `Branched`, the check box would be disabled.

Floating Shares

In general, floating shares are considered bad practice, especially for items that branch (folders, files, and change requests) because synchronization and propagation can happen automatically and may be unwanted.

- Changes from the parent item will automatically propagate to a child item if the child item’s configuration is set to float. You can see this in the reference tree for either item. This behavior happens regardless of whether the **Branch On Change** option is selected or not. Once the child item branches, the floating stops. In order to branch, the item’s behavior must be set to **Branch On Change.**
- Changes from the child item will automatically propagate to the parent only if the child item’s configuration is set to float and the ‘branch on change’ option is not selected.

For floating folders, there are additional consequences:

- New items added to the parent folder can float into the child folder until the child folder branches.
- New items added to the child folder can float into the parent folder if the child folder’s configuration is set to float and the ‘branch on change’ option is not selected.

Notice that floating can occur uni-directionally (parent reference to child reference) or bi-directionally (parent to child and child to parent). Also notice that a floating folder affects what happens to items newly added to it. As a result of floating, the parent and child items can be identical. Actually, they are the same item because no branching has occurred.

When this happens, a View Compare/Merge session reports the difference type **Unchanged**. This can be surprising if you know that you have changed the item recently, or if you compare labels within a view and see that this particular item has changed since the label was made. Understand that **Unchanged** can mean **Same item in both locations** in addition to **Unchanged**.

While floating is generally considered a bad practice, this feature is allowed because it can be useful in certain circumstances, particularly for items that cannot branch (requirements, tasks, and topics). When items that cannot branch have a pinned (non-floating) configuration, they are also read-only.

Note: An item that can branch is read-only if it is both pinned and set to branch on change. This is because the item cannot branch and the change cannot float to another location.

Some customers put items that cannot branch in a floating folder so that the can float from one view to another. They also set each item’s configuration to floating so that it can be changed from any view in which it can be seen. Those changes, in turn, can be seen in all views where the item is visible.

VCM has a few of ways of finding floating items. It looks for the following item difference types:

- Floating Child Share (Row 1210 in the Action Decision Table)
- Floating Child Share, Source On Root Branch (Promote) (Row 1230)
- Target Folder Has Floating Share In Source (Row 200)

All of these are discussed in the topic *Understanding VCM Difference Types*. Difference types 1210 and 1230 only occur if the VCM option **Fix Floating Child Shares** is used in the session. VCM can correct for these differences. When difference type 200 occurs, it results in a **Fail** condition. To commit the VCM session, a manual change must be made to the floating folder in the target view, followed by a refresh of the view. Then the VCM session must be redone. It might be wise to check other foldersin the problem folder's reference tree. VCM only detects the problem between the two views being compared in the VCM session. There may be other floating references.

Floating Items in a VCM Session

VCM has a few of ways of finding floating items It looks for the following item difference types:

- Floating Child Share (Row 1210 in the Action Decision Table)
- Floating Child Share, Source On Root Branch (Promote) (Row 1230)
- Target Folder Has Floating Share In Source (Row 200)

All of these are discussed in the topic “Understanding VCM Difference Types”.

Difference types 1210 and 1230 only occur if the VCM option **Fix Floating Child Shares** is used in the session. VCM can correct for these differences. When difference type 200 occurs, it results in a **Fail** condition. To commit the VCM session, a manual change must be made to the floating folder in the target view, followed by a refresh of the view. Then the VCM session must be redone. It might be wise to check other folders in the problem folder's reference tree. VCM only detects the problem between the two views being compared in the VCM session. There may be other floating references.

Hiding Folders and Files

Using the **Folder Properties** dialog box, you can set the **Visible** property to exclude folders and their files from visibility.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Double-click the folder. The **Folder Properties** dialog box opens.
3. Select the **Name** tab and uncheck the **Visible** option. This hides the folder and the files it contains
4. Click **OK**.

Note: To make the folder visible again, check the **Visible** option in the **Folder Properties** dialog box.

Highlighting Items of Interest

To highlight specific items (except audit entries) on the upper pane, you can add a flag to them. For example, you might wish to flag items related to a particular customer request.

Flags are set, viewed, and removed by the user who created them. If an item has been flagged, the **Flag** field displays **Yes**. If an item is not flagged, the **Flag** field displays **No**.

To display flagged items, the upper pane must be in list format.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. You can select an item on any of the component tabs in the upper pane except the **Audit** tab.
3. Select the item you want to flag.
4. Choose **Flag** from the corresponding item menu or the context menu.
5. If the items in the upper pane appear in tree format, switch to the list format to display the flagged items.

Note: If the **Flag** field does not appear, right-click a column header, and choose **Show Fields** from the context menu. Select the **Flag** field from the **Available Fields** list, click **Add**, then click **OK**.

Note: Click Remove Flat to remove a flag.

Tip: Use the following shortcut keys: **Ctrl+F2** to flag an item. Use **Ctrl+Shift+F2** to remove a flag from an item.

Linking Items Internally or Externally

This procedure describes how to link two items, either internally in the same server configuration, or linking between two items located on different server configurations, called *external linking*.

In StarTeam, an *item* is a file, change request, requirement, task, or topic. A link is a connection between two folders, two items, or a folder and an item on the same server, or on two different servers (called *External Links*).

Creating links can be quite useful. For example, linking a file to a change request allows you to mark it as fixed when you check in the edited file. By linking files to the requirements document that the files fulfill, you can easily refer to or update the document.

You can create several links at the same time if you want to link several items of the same type to one particular item. For example, you might wish to link several change requests to a single file. To accomplish this, you can create links using the **Folder Tree** menu, component menu, context menu, or **Link** button on the toolbar.

Note: When you create external links between items on different server configurations, both server configurations need to be opened in the StarTeam Cross-Platform Client to be able to create or view the external links. Also, to create external links, you must have the access rights to the generic external link access rights, such as create, see, modify, delete.

1. Begin the link process by doing one of the following:
   - Open the server configuration and view which contains the items you want to link. If you want to link two items on different server configurations, open both server configurations and views in the StarTeam Cross-Platform Client.
   - Select a folder from the folder tree or in the upper pane on the **Folder** tab.
   - Click a component tab in the upper pane, such as **File**, **Change Request**, **Requirement**, **Topic**, or **Task**, and select one or more items.

   Note: You cannot link two folders to each other with the method described in this procedure.
2. Right-click the selected item(s) on the component tab and choose **Links** > **Create Link** . The **Links** menu is also available on the component menu that corresponds with the selected component tab.

   This action changes the mouse pointer and displays it as a knotted rope.

   Note: If you initially select an item from the upper pane, you can also use the **Link** button on the toolbar. However, this button is disabled if you start the link with a folder.

3. Select the folder or item(s) for the end of the link in the project on the other StarTeam Server. This can be:

   - A StarTeam folder (if you have not already selected a folder).
   - One or more other files.
   - One or more change requests or change packages.
   - One or more requirements.
   - One or more topics/responses.
   - One or more tasks/subtasks.

   To locate all items, you may need to switch to a different component tab or use the **All Descendants** button on the toolbar.
4. Click **Folder Tree** > **Links** > **Complete Links** or click the **Link** button again on the toolbar.

   This button is disabled if you are linking an item to a folder.
5. Verify that the links exist by doing one of the following:

   - Select a linked item, then click the **Link** tab on the lower pane to view the links for the item.
   - Right-click a linked folder, then choose **Properties** to display the **Folder Properties** dialog box. Click the **Link** tab to view the link. (The **Link** tab will not appear in this dialog box if you do not have access rights to view links.)

You can also view a link by selecting either of its ends. The end you select, whether a folder or an item, is called the source. The other end of the link is called the target and is listed in the **Item Type** column on the **Link** pane.

Tip: If you change your mind about creating a link after you have started to create it, but before you have finished completing it, you can select **Links** > **Cancel Link** on the **Folder Tree** menu, the component menu, or the context menu. If you are using the **Link** button on the toolbar, press **ESC**.

Note: External links can also be created using drag-drop. With both views open, select the source item, press **CTRL + SHIFT**, then drag-drop it on the target item.

Locking and Unlocking Items

Before changing the contents of a file or editing item properties, you should exclusively lock the file or item. This action informs other team members that you intend to make changes. Files, change requests, requirements, tasks, and topics can all be locked.

Exclusively locking an item prevents others from creating new revisions of it before the lock has been released. You can lock and unlock any type of item as a separate operation. In addition, you can lock and unlock files as part of the check-in and check-out processes.

If an item is exclusively locked by someone else, you can review its properties but cannot change them. Normally the words **Read Only** and the name of the user who has locked the item will appear on the title bar.

Locking an Item Using the Toolbar

1. Select one or more items in the upper pane.
2. Click the **Lock** button on the toolbar. The selected items become exclusively locked, and you are listed as the user who has locked them.

Tip: To unlock an item, select the locked item and click **Unlock** on the toolbar.

Locking an Item Using the Menu

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.

2. You can select an item on any of the component tabs in the upper pane except the **Audit** tab.
3. Right-click an item and choose **Lock/Unlock**. This opens the **Set My Lock Status** dialog box.
4. Select a lock status option:

Unlocked	Removes your exclusive or non-exclusive lock on the selected items.
Exclusive	Prevents others from creating new revision of this item (until you release the lock or another person breaks your lock).
Non-exclusive	Indicates that you are working on the item and may possibly make changes (not recommended for items other than files).

5. Optionally, check **Break Existing Lock** to break another team member's lock on the item.

   If e-mail is enabled, StarTeam will send an e-mail message to the team member whose lock has been broken to inform him or her of this fact.

   Note: You must be granted the appropriate privileges to be able to break another person's locks.

Tip: To unlock an item, select the locked item and click **Unlock** on the toolbar.

Marking Items Read or Unread

Change requests, requirements, tasks, and topics that you have not read display in boldface type if you are the user responsible for the item, or if you are the recipient of the topic. After you have reviewed the item properties, the boldface type is replaced by regular type.

To reread all the items on a certain subject, it is sometimes convenient to mark all of the items unread to ensure that you do not miss any. You can also mark an entire requirement, task, or topic tree as unread.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click the **Change Request**, **Requirement**, **Task**, or **Topic** tab.
3. Select one or more items that you want to make bold (unread) or not bold (read).
4. Right-click the selected items and choose **Mark As Unread** or **Mark As Read**.

   The items are marked as specified, and the type style changes accordingly.

Moving Folders or Items

Folders and items, such as files (including Not in View files) and change requests, can be moved from one project view to another as long as the two views are on the same server configuration. Moving a folder also moves its contents, its child folders, and their contents. When an item is moved to another project view, it belongs to the new view, although its behavior, configuration, and other properties do not change. It loses any labels it had in the previous view, however, because labels cannot move from view to view. Also, if you roll back the view to an earlier point in time, you will no longer see the folders and/or items that have been moved.

Moving a folder or item within a view causes that folder or item to be copied in that view's child or parent views, if branching has not occurred. In this application, a move is a copy operation followed by a delete operation, and delete operations are not propagated from view to view for folders and items that have not branched. Therefore, the view in which the move was made has one copy of the folder or item in the new location, while the related views have two copies of the folder or item, one in the original location and one in the new location, the equivalent of a share.

Note: When you move a folder or an item, the access rights set at the folder or item level accompany it. Also, in some cases, moving a folder or item to another view enables its disabled **Branch on Change** check box.

Moving a Folder or Item Within the Same View

1. Choose **Project** > **Open** . The **Project View** window opens.
2. Drag the folder or item that is to be moved from one location in the view to another.

   Tip: You can optionally move the working folder/file using this dialog box. This option is automatically selected. If you do not want to move the working folder or file, clear the check box.
3. A message box appears asking you to confirm.
4. Click **Yes**.

Moving a Folder or Item Between Two Different Views

1. Choose **Project** > **Open** . The **Project View** window opens.
2. Choose **Project** > **Open** to open a second project in another window.
3. Make sure both project views are visible. Use one of the **Window** menu commands (**Cascade**, **Tile Vertically** or **Tile Horizontally**) to this, and resize the windows if required.
4. Drag the folder or item from one view to the other.
5. A message box appears asking you to confirm.
6. Click **Yes**.

Caution: You cannot move tasks and sub-tasks that have been exported from Microsoft Project to StarTeam.

Restoring Folder Selection on Tab Change

When you click a new component tab in the upper right pane, StarTeam gives you the option of retaining the folder currently selected from the folder hierarchy or replacing it with the root folder.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Select the **Workspace** tab.
3. Do one of the following:
   - Check **Restore Folder Selection on Tab Change** to retain the currently selected folder when the component tab changes.
   - Uncheck **Restore Folder Selection on Tab Change** to select the root folder every time the component tab changes
4. Click **OK**.

Selecting Referenced Items in Other Views

The **Reference** tab located in the lower pane provides a context menu containing commands for viewing the properties of an item and for opening a referenced item in a different view.

1. Select an item in the upper pane.
2. Select the **Reference** tab in the lower pane.
3. Right-click the reference for the item, and choose **Select Referenced Item** from the context menu.

   The view containing the item opens (in addition to any views that are already open) with the item selected. If the referenced view is already open, the StarTeam Cross-Platform Client opens an additional copy of the view so that it does not disturb anything that may have already been set up within that view.

Sharing Folders or Items

You can share folders and items, such as files or change requests, between views if the views belong to projects that are located on the same server configuration. You can also share a folder or item at two locations in the same view.

Branching a view negates all shares, not just the ones between parent and child views.

Keep in mind the following points about sharing folders or items:

- When you branch a view, any manual shares between items in the same view are not retained in the view's child view.
- When a folder is shared, users of both views can access its contents, including its child folders and their contents.
- As a rule, the behavior of the shared item is governed by the **Set items Shared Into View to Branch on Change** property of the new view, not by that of its parent view. If the new view is a reference view (that is, a view that does not permit branching), it does not have that property, so the shared item's Branch on change setting is controlled by the setting of its parent view.
- The "branch on change" behavior of a shared item is specific to the folder it is in and the **Branch on change** check box is selected by default for the shared file.
- Tasks and topics do not have branching behavior so this view property does not affect them.
- If a folder or item is shared, its configuration (floating, based on a label, a promotion state, or a point in time) is initially identical in both views. However, the configuration can be modified in either view, which means that shared items can differ. To freeze shared folders or items, you must configure each of them to a specific date and time. Once items have been shared into a view, you can change their branching behavior on an item-by-item basis.
- Shared folders or items have the access rights originally set for them at the folder or item level, until they branch in their new location. Branching creates a new object that initially has no access rights at the folder or item levels.
- The shared folder or item loses any labels it had in the previous view. Labels cannot be moved from view to view. However, the shared folder or item will have all the labels you attach to them in their new location regardless of whether they branch or not.

Sharing a Folder or Item in Two Locations in the Same View

1. Choose **Project** > **Open** . The **Project View** window opens.
2. Press **Ctrl** and drag the folder or item that is to be shared from one location in the view to another. A message box appears asking you to confirm.
3. Click **Yes**.

Sharing a Folder or Item Between Two Different Views

1. Choose **Project** > **Open** . The **Project View** window opens.
2. Choose **Project** > **Open** to open a second project in another window.
3. Make sure both project views are visible. Use one of the **Window** menu commands (**Cascade**, **Tile Vertically** or **Tile Horizontally**) to this, and resize the windows if required.
4. Press **Ctrl** and drag the folder or item from one view to the other view. A message box appears asking you to confirm.
5. Click **Yes**.

Caution: When you share tasks that have been exported from Microsoft Project, you must share an entire task tree, starting with the root task.

Viewing or Modifying Item Properties

This section explains how to use the standard properties dialog to edit item properties. Depending on how your team has set up the application, you may see a totally different dialog box called an alternate property editor (APE).

Every time the properties of an item are modified, a new revision of that item is created. If you modify a property, you should also create a revision comment explaining the modification using the **Revision Comment** tab.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Do one of the following:
   - Right-click the folder or item and choose **Properties**.
   - Right-click an item on the **Reference** tab in the lower pane and choose **Item Properties**.
   - Click the **Properties** button.
3. Modify any of the property fields in the corresponding **Properties** dialog box that opens, then click **OK**.

Tip: You can force Java to not cache Microsoft Windows file properties by placing the following option in the CPC stjava file: `-Dsun.io.useCanonCaches=false`.

Embedded Property Editors

Embedded property editors allow you to:

- View, edit, and compare multiple StarTeam items at once.
- Copy values from one property editor to another.
- Continue to do other work while the embedded property editor is open, even if it has unsaved changes.

Note: StarTeam supports embedded editors for any combination of Alternative Property Editors (APEs), Layout Designer forms, and default property editors. However, embedded editing is not supported for APEs that do not implement the `AbstractItemEditor` interface; these APEs will be displayed in a property editor dialog.

Using Embedded Property Editors

1. Click the **Use Embedded Editor** button. The persistent default display mode for the selected item type will now be embedded editors instead of property editor dialogs.
2. Do one of the following:
   - Double-click the item.
   - Right-click the item and select **Properties** from the shortcut menu.

Each item opened in an embedded editor is displayed as a tab in the **Embedded Editor** pane below the **Items** pane. Each tab represents a unique item (and version) of the same or different Item type. You can view multiple versions of the same Item as well as Items of different types in the same Embedded Editor pane.

Once an embedded editor is open, use the **Details Display** and **Embedded Editor Display** icons to toggle back and forth between the Details pane and the Embedded Editor pane.

When an item is edited, asterisks will appear in the title bar and tab of the item's embedded property editor to indicate there are unsaved changes. On the right side of the title bar, all of the toolbar icons will also be enabled (the **Close** and**Close All** icons are always enabled). Each item's embedded property editor toolbar contains the following icons:

The **Compare Properties** icon is enabled when there are unsaved changes to the item displayed in the embedded property editor. Click the icon to view the Compare Properties dialog and review changes made to the item during the current session. The original and new values for edited fields are displayed in bold type. Note that this icon is currently not available for an embedded APE editor.

The **Save and Close** icon is enabled when there are unsaved changes to the item displayed in the embedded property editor. Click the icon to save the changes to the Item and close the item's embedded editor tab.

The **Apply** icon is enabled when there are unsaved changes to the item displayed in the embedded property editor. Click the icon to save the changes to the Item and leave the item's embedded editor tab open.

The **Reset** icon enabled when there are unsaved changes to the item displayed in the embedded property editor. Click the icon to discard changes made to the item during the current session and leave the item's embedded editor tab open.

The **Close** icon is always enabled. Click this icon to close the item displayed in the embedded property editor. You will be prompted to save any changes to the item. You can also click the **X** on the tab to close the item.

The **Close All** icon is always enabled. Click this icon to close all open embedded editors. You will be prompted to individually save each item that has unsaved changes.

Dragging and Dropping Items into the Details/Embedded Editor Pane to View an Embedded Editor

1. Drag the first item(s) from the **Items** pane into the **Details/Embedded Editor** pane.
2. Drag any successive item(s) to the title bar of the active item's embedded editor tab.

Note: An item cannot be dropped onto an enabled toolbar on the right side of the title bar.

You can drag and drop an item into the **Details/Embedded Editor** pane whether or not embedded editors is the current display mode for the selected item type.

Toggling Full Screen and Standard View

1. Double-click the title bar of the embedded property editor to view it in full screen mode.
2. Double-click the title bar of the property editor again to return to standard view.

Note: If you resize the **Details/Embedded Editor** pane, the client will automatically display the split pane bars in the same position the next time you open the view window. This position can vary depending on whether you are viewing Details or Embedded Editors.

Granting Item-Level Access Rights

Although access rights can be set on individual items, this is rarely done. For example, if you really need to allow only one person to know about a particular file, you can give only that person access rights to that file. However, by default, the owner of the file and anyone belonging to a group with the correct privileges can still see that file.

To ensure that only that one person can access the file, you would have to stop the StarTeam Server from checking for privileges. Then the access to every object would be controlled solely by access rights.

Like folders, items in a child view retain the access rights they had in the parent view until they branch into new objects. Items lose their access rights only when branching.

Moving Folders or Items	When you move a folder or an item, the access rights set for it at the folder or item level go with it. It also has the same behavior, which either allows it or stops it from branching on change.
Sharing Folders or Items	When you share a folder or item, the access rights set for it at the folder or item level accompany it, until the folder or item branches. When you share a folder or item, its behavior may change. When shared, the behavior immediately becomes able to branch on change, even if the **Branch On Change** check box was disabled in the original location. Whether the **Branch On Change** check box is selected or cleared depends on the property setting for the destination view **Set Items That Are Shared Into View To Branch On Change**. When the folder or item branches in its new location, a new object is created in the repository, and that new object initially has no access rights at the folder or item level.

Generic Item Access Rights

The following table describes the access rights that are available from the **File** nodes in the **Project Access Rights**, **View Access Rights**, **Folder Access Rights**, and **File Access Rights** dialog boxes.

Generic item rights

See item and its properties	See files in the files list (upper pane) and view file properties by selecting **File** > **Properties** .
Modify properties	Change the file properties. Modifiable properties include the archive/file name, description, executable bit setting (useful only for non-Microsoft Windows platforms), compression, storage options, and custom properties. If used, an alternate property editor (APE) may restrict the properties that can be modified and the users who can modify them still further.
Delete from folder	Delete files from their folders.
Change item access rights	Change access rights for the files. If you change this setting, be sure that you remain one of the users who can change access rights.
See history	See file history in the history pane.
Perform maintenance	Change the revision comments for past revisions.
Set exclusive locks	Lock files exclusively.
Break exclusive locks	Remove someone else’s exclusive lock on the files.

Links: Internal and External

A link is a connection between two folders, two items, or a folder and an item on the same server, or on two different servers (called *External Links*). You can view links by using the **Link** tab in the lower pane of the StarTeam Cross-Platform Client.Creating links can be quite useful. For example, linking a file to a change request allows you to mark it as fixed when you check in the edited file. By linking files to the requirements document that the files fulfill, you can easily refer to or update the document.

In addition, linking files to change requests enables you to mark the change requests as fixed when you check in the corresponding files. In turn, if you link each set of files to the requirements document that the files fulfill, you can easily refer to or update the document.

A link does not provide a connection to a single share (or reference), but to all related shares and branches of an item. Links are not affected by any item operations, such as branching, moving, sharing, and so on. By default, a link connects the tip revisions of the linked pair.

Links can either be pinned or floating:

Pinned You can lock the link to the tip revision. The context menu in the link tab enables you to pin links to the source or target items or both.

Floating You allow the link source or target to change from tip revision to tip revision as new revisions are created. The context menu in the link tab enables you to float links to the source or target items or both.

Links, as with all other items, have context menus in their tabbed panes which allow you access to more information about the item.

Linking Items Internally or Externally

This procedure describes how to link two items, either internally in the same server configuration, or linking between two items located on different server configurations, called *external linking*.

In StarTeam, an *item* is a file, change request, requirement, task, or topic. A link is a connection between two folders, two items, or a folder and an item on the same server, or on two different servers (called *External Links*).

Creating links can be quite useful. For example, linking a file to a change request allows you to mark it as fixed when you check in the edited file. By linking files to the requirements document that the files fulfill, you can easily refer to or update the document.

You can create several links at the same time if you want to link several items of the same type to one particular item. For example, you might wish to link several change requests to a single file. To accomplish this, you can create links using the **Folder Tree** menu, component menu, context menu, or **Link** button on the toolbar.

Note: When you create external links between items on different server configurations, both server configurations need to be opened in the StarTeam Cross-Platform Client to be able to create or view the external links. Also, to create external links, you must have the access rights to the generic external link access rights, such as create, see, modify, delete.

1. Begin the link process by doing one of the following:
   - Open the server configuration and view which contains the items you want to link. If you want to link two items on different server configurations, open both server configurations and views in the StarTeam Cross-Platform Client.
   - Select a folder from the folder tree or in the upper pane on the **Folder** tab.
   - Click a component tab in the upper pane, such as **File**, **Change Request**, **Requirement**, **Topic**, or **Task**, and select one or more items.

   Note: You cannot link two folders to each other with the method described in this procedure.
2. Right-click the selected item(s) on the component tab and choose **Links** > **Create Link** . The **Links** menu is also available on the component menu that corresponds with the selected component tab.

   This action changes the mouse pointer and displays it as a knotted rope.

   Note: If you initially select an item from the upper pane, you can also use the **Link** button on the toolbar. However, this button is disabled if you start the link with a folder.
3. Select the folder or item(s) for the end of the link in the project on the other StarTeam Server. This can be:
   - A StarTeam folder (if you have not already selected a folder).

- One or more other files.
- One or more change requests or change packages.
- One or more requirements.
- One or more topics/responses.
- One or more tasks/subtasks.

To locate all items, you may need to switch to a different component tab or use the **All Descendants** button on the toolbar.

4. Click **Folder Tree** > **Links** > **Complete Links** or click the **Link** button again on the toolbar.

   This button is disabled if you are linking an item to a folder.
5. Verify that the links exist by doing one of the following:
   - Select a linked item, then click the **Link** tab on the lower pane to view the links for the item.
   - Right-click a linked folder, then choose **Properties** to display the **Folder Properties** dialog box. Click the **Link** tab to view the link. (The **Link** tab will not appear in this dialog box if you do not have access rights to view links.)

You can also view a link by selecting either of its ends. The end you select, whether a folder or an item, is called the source. The other end of the link is called the target and is listed in the **Item Type** column on the **Link** pane.

Tip: If you change your mind about creating a link after you have started to create it, but before you have finished completing it, you can select **Links** > **Cancel Link** on the **Folder Tree** menu, the component menu, or the context menu. If you are using the **Link** button on the toolbar, press **ESC**.

Note: External links can also be created using drag-drop. With both views open, select the source item, press **CTRL + SHIFT**, then drag-drop it on the target item.

Deleting Links

You can delete an existing link from either the source end or the target end of the link.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.
3. Select one or more links to delete.
4. Right-click the selected links and choose **Links** > **Delete Link** .

Linking Files to Process Items

If process rules are enforced for a project, linking and pinning new file revisions to a process item is required. Otherwise, this step is optional, and you can select any change request, requirement, or task as a process item.

Linking and Pinning a File Revision to a Process Item

1. Click the **File** tab.
2. Select one or more files.
3. Click **File** > **Check In** . The **Check In** dialog box opens.
4. Check **Link and Pin Process Item**.
5. Click **Select**. The **Select Process Item** dialog box opens.
6. Select one of the following to limit the list of possible process items:

List All Permitted Items Displays all items that can be used as process items. If process rules are not enforced, the list contains all change requests, requirements, and tasks.

List All Permitted Items Assigned To Me Displays all the items for which you are responsible that can be used as process items.

List Linked Items Lists the process items that are already linked to at least one of the files you are checking in. No process item appears on the list more than once, even if it is linked to several files. Also, when a process item is linked to more than one file, the dialog box displays the name of only one file. Despite this fact, the application will update or create links for every file being added.

7. Select the **Change Request**, **Requirement**, **Task**, or **Custom Component** tab to restrict the list to a specific type of item.
8. Select one item in the list to be the active process item and click **OK**.
9. Optionally, in the **Check In** dialog box, check **Mark Selected Item As Fixed/Finished/Complete** if work on the process item is completed.
10. Complete filling in the fields in the **Check In** dialog box and click **OK**.

Tip: If process rules are enforced, the use of some change requests, requirements, or tasks as process items may not be permitted because of their status. If you select such an item and click **OK**, the application notifies you of this fact. By double-clicking the item in the list box, you can display its properties and change it to a permitted status for a linked process item.

Linking and Pinning a File Revision to the Active Process Item

1. Click the **File** tab.
2. Select one or more files.
3. Click **File** > **Check In** . The **Check In** dialog box opens.

   If you have already specified a process item, it will appear by default in the **Link and Pin Process Item** field.
4. Check **Link and Pin Process Item**.
5. Optionally, in the **Check In** dialog box, check **Mark Selected Item As Fixed/Finished/Complete** if work on the process item is completed.

   Note: Selecting this option will change the value of the process item's status property to the `Closed` state specified on the **Project Properties** dialog box.
6. Complete filling in the fields in the **Check In** dialog box and click **OK**.

Linking Specific Revisions

Each end of a link has an associated start revision and an end revision that determines the range of revisions to which the link applies. The start revision is always fixed at the time of the creation of the link and is set to the first revision on the current branch. The end revision is under the user's control and may be fixed (or pinned), which puts an upper bound on the linked revisions, or floating, which does not. If a link end is pinned, it is always attached to the same version of the linked folder or file. If a link end floats, it moves from revision to revision, as new revisions of the linked folder or item are created.

By default, a link connects the tip revisions of the linked pair. The revisions selected for both links appear as columns on the **Link** pane.

Determining whether a link is visible on a given item is simple. If any of the revisions between the start and the end revision defined for the link are in the history of the selected item, it is visible. Otherwise, it is not.

Linking to a Tip Revision

To link to a tip revision:

1. Right-click a folder or item for which you have created a link.
2. Do one of the following:
   - If you selected an item, click the **Link** tab in the lower pane.
   - If you selected a folder in the **Folder Tree**, choose **Folder Tree** > **Properties** to display the **Folder Properties** dialog box, and click the **Link** tab.
3. Select one or more links in the **Link** pane.
4. Right-click the selected link(s) and choose one of the following options:

Pin Link > To Source Item At Tip	This command pins the link to the tip revision of the source (that is, the folder whose properties you are reviewing or the item selected from the upper pane).
Pin Link > To Target Items At Tip	This command pins the link to the tip revision of the target (that is, the folder or item in the Item column on the Link pane).
Pin Link > To Source And Target Items At Tip	This command pins the link to the tip revisions of both the source and target.
Float Link > To Source Item	This command allows the link to float from tip revision to tip revision of the source as new revisions are created.
Float Link > To Target Items	This command allows the link to float from tip revision to tip revision of the target as new revisions are created.
Float Link > To Source And Target Items	This command allows both the source and the target of the link to float from tip revision to tip revision.

Linking to a Specific Revision

To link to a specific revision:

1. Right-click a folder or item for which you have created a link.
2. Do one of the following:
   - If you selected an item, click the **Link** tab in the lower pane.
   - If you selected a folder in the **Folder Tree**, choose **Folder Tree** > **Properties** to display the **Folder Properties** dialog box, and click the **Link** tab.
3. Select a link.
4. Right-click the selected link and choose **Link Properties**. The **Link Properties** dialog box opens.
5. Optionally, type a description or comment about the link in the **Description** or **Comment** field. This text will appear in the **Comment** or **Description** column of the **Link** pane.
6. Do one of the following in the **Source Item** group:
   - Click **Pin**: Displays the **Select Version** dialog box. Select a specific folder or item revision from the list. This revision number will appear in the **Selection Version** column of the **Link** pane.
   - Click **Float**: The link is always connected to the tip revision of this item.
7. Do one of the following in the **Is Linked to Target Item** group:
   - Click **Pin**: Displays the **Item Version** dialog box. Select a specific folder or item revision from the list. This revision number will appear in the **Item Version** column of the **Link** pane.
   - Click **Float**: The link is always connected to the tip revision of this item.
8. Click **OK**.

Note: You can link items from a project view on one server to an item in another project or view on a different server. This is called an external link.

Viewing Links

StarTeam displays links at the bottom of the client in the **Link** pane for any item selected item in upper pane.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.

You can control whether to display all links, or only enhanced links by selecting the appropriate option at the top of the **Link** pane.

You can also use the list in the **Link** pane (called **Linked Item**) that identifies the item for which links are displayed. Initially, the item displayed is the one selected in the upper pane. To select a different linked item, click the arrow beside the **Linked Item** list box

Note: You can drag any link displayed in the Link tab to the **Linked Item** list box. This action changes the linked item as well as the links displayed in the **Link** pane. This feature is especially helpful in viewing the links to change packages. If you switch to a different tab and then return, the list box reverts to displaying only the item selected from the upper pane. When you select a process item in the upper pane, the only link displayed in the **Link** pane may be the link to a change package. To see the items that are linked to the change package (and are therefore linked indirectly to your process item), drag the change package link from the list in the **Link** pane onto the **Linked Item** list box. The **Linked Item** list box now displays the change package, and the **Link** pane list now displays the change package links. For example, the **Link** pane could show a link to each file checked in using the previously selected process item, as well as a link to the process item itself.

Link Tab

The **Link** tab displays all the links associated with an item selected in the upper pane. Radio buttons also let you specify whether to show all types of links, or just the enhanced process links.

Each link has two ends: a source and a target. The source is the item selected from the upper pane or from the folder hierarchy (if you are viewing the **Link** tab of the **Folder Properties** dialog). Each row on the **Link** tab defines a link that has the source item or folder as one of its ends. Columns in the link list identify the other end of the link for you.

The following contains the columns provided in the **Link** tab. These columns also appear on the **Link** tab of the **Folder Properties** dialog. (The **Folder Properties** dialog has no **Link** tab if you do not have the access rights that allow you to view links.)

Created By
The name of the user who created the link.

Created On
The date/time the link was created.

Type
The type of link that was created. For example: `Manual` or `Copy`.

View
The assumption is that you want to locate items in the current view whenever possible, so the view field contains one of the following:

- The name of the current view if the link was created in the current view, or if the link was created in another view but a shared copy of the item (to which the current item is linked) exists in the current view.
- The name of the view containing the link. If the view is a reference view, this is the name of the parent view.

Folder
The name of the folder in which the folder or item in the link list resides.

Item Type Identifies the type of item the target end of the link is attached to. This item is the one listed in the link list. The values that can appear in this column are Folder, File, Change Request, Requirement, Task, and Topic.

Item Identifies the item that the target end of the link is attached to by its folder name, filename, change request number, task number, and so on.

Item Details Describes the item, using a folder description, file description, change request synopsis, requirement name, task name, or topic title.

Item Version When the ends of a link are pinned to specific revisions of the folders or items, those revisions appear in the **Item Version** and the **Selection Version** columns of the link list (if that revision is in the current view).

Selection Version The **Item Version** displays the version number of the target end of the link if that revision is in the current view.

The **Selection Version** displays the revision number of the source end of the link. The source is either the folder selected from the folder hierarchy (when viewing a link for a folder) or the item selected from the upper pane (when viewing the link for an item).

When no revision number is in the column, that end of the link is floating rather than pinned.

Comment From the **Link Properties** dialog box, you can enter a comment about this particular link. That comment appears in this column.

Description From the **Link Properties** dialog box, you can enter a description about this particular link. That description appears in this column.

File Status Displays the status of the file based on the operations performed on the file.

Locked By If the file is locked, the column displays the name of the user who has locked the file.

Folder Path Shows folder path information only when the linked item is in the same view. Otherwise, displays the following message: `Unavailable. Item in another view.`

Checking Linked Files In and Out

If you are checking in a file that has one or more linked change requests, you can do this from the **Link** pane.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.
3. Select one or more files in the **Link** pane.
4. Right-click the selected item in the upper pane or the selected files in the lower pane.
5. Do one of the following:
   - **Linked Files** > **Check In All**
   - **Linked Files** > **Check Out All**
6. Use the **Check In** or **Check Out** dialog box as you normally would to check files in our out.

Customizing Link Item Properties

You can view or modify folder and item properties directly from the **Link** pane.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.

3. Right-click a link and choose **Item Properties**. The **Properties** dialog box opens. It displays information about both the Source and Target Items.

Customizing Link Properties

You can view or modify link properties from the **Link** pane.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.
3. Right-click a link in the **Link** pane and choose **Link Properties**.
4. The **Link Properties** dialog box opens where you can view or modify certain properties or add a description or comment.

External Link Access Rights

External Link item access rights available from the **Project** and **View** menu **Access Rights** menu item.

Generic external link rights

Create external link	Allows the specified users and groups to create an external link between items in two different views, or on two different servers.
See external link and its properties	Allows the specified users and groups to view the external link and its properties.
Modify properties	Allows the specified users and groups to modify the external link properties. To view the properties of an external link, open both views which contain the link. In one of the views. Click on the **Link** tab, then right-click the external link and choose **Link Properties**. You can only modify the description property.
Delete from view	Allows the specified users and groups to delete an external link from the view. This is an irreversible action.

Change Requests

The *change request* component provides a defect tracking system that allows you to record defects in products, projects, or services and suggest possible enhancements. A change request is a request to change something within the scope of a project. For example, you might suggest a product enhancement or request a fix for an error or problem. To use the change request tracking system effectively, you need to understand the model on which it is based.

The change request component allows you to:

- Attach change requests to any folder. In the application, change requests can be attached to any project folder or shared among folders or other views in the same server configuration. You can also link a change request to any other item, such as a file. In many other defect tracking systems, a change request can be associated only with a project, even though it requires modification of a particular file.
- Save time when updating change requests. When you check in a file or group of files, you can indicate the change requests that are fixed by the files being checked in. This feature saves the time required to change the status of each change request separately.
- Make only appropriate status changes. When you create a change request, the status options are `New`, `Open`, `Deferred` or a resolution. The resolutions are `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, and `Is Duplicate`. After resolution, a change request can only be verified or reopened. After verification, a change request can only be closed or reopened.

- Benefit from automatic changes based on the status of the change request. The application automatically changes the person responsible to coincide with the current status of the change request. When a change request is resolved, the responsibility for the change request automatically reverts to the person who entered the change request, who is usually the best person to verify its resolution. When a change request is reopened after being resolved, the responsibility is automatically set to the user who resolved it. If desired, you can override these automatic changes and make another person responsible.
- Base change requests on the build in which the change request is resolved. When a change request receives a **Fixed** or **Documented** status, the value of its **Addressed In Build** field becomes **Next Build**. When that build label is created, the application replaces **Next Build** with the name of the build label, letting testers know the build to use when verifying change requests.

Note: This help system explains how to use the standard property dialog to create and edit change requests. Depending on how your team has set up the application, you may see a different dialog called an alternate property editor (APE). Even if you use the standard property dialog for change requests, your company or team leader may implement change request guidelines that differ from those discussed in this help system.

Creating Change Requests

Development teams create change requests to record problems and enhancement requests and track their resolution or implementation. Many teams have adopted specific guidelines that govern the creation and content of change requests, for example, instructions about what can be entered in the **Component** and **Category** fields. Be sure to follow these guidelines, if they exist.

1. Define the change request summary information on the **Synopsis** tab.
2. Describe the change request on the **Description** tab.
3. Optionally, specify a temporary workaround on the **Solution** tab.
4. Optionally, specify custom change request properties on the **Custom** tab.
5. Optionally, attach files related to this change request on the **Attachments** tab.
6. Optionally, add comments about the change request on the **Comment** tab.
7. Click **OK**. StarTeam assigns a unique number to the change request and displays the summary information.

Working with Attachments

You can attach image files, HTML/plain text formatted notes, and other file types to a component using the **Attachments** tab on the **Properties** dialog box of the selected component. The HTML/plain text attachment is similar to a note in which you can easily format and align text without having to use an external editor.

Note: Attachments can only be used with Change Request, Requirement, Task, and Topic components.

1. Select a folder from the folder hierarchy and click the component tab in the upper pane (for example, **Change Request**) .
2. Open a component item that you want to add an attachment to, delete an attachment, or edit an existing attachment.
3. Click the **Attachments** tab.

To add an attachment

1. Click **Add**. The **Open** dialog box displays.
2. Select the file you want to attach to the component and click **Open**. To select multiple items, use the keyboard functions **Shift+Click** or **Ctrl+Click.** The selected item appears in the **Attachments** list on the **Attachments** page of the **Properties** dialog box.

3. Click **OK**.

To add an HTML or plain text attachment

1. On the **Attachments** tab, click **Add HTML**. The **Attachment Editor** dialog box displays.
2. In **Name**, type a descriptive file name or title to identify the attachment.
3. In the text box, enter descriptive content for the body of the attachment and apply the formatting of your choice. You can also insert an image in the attachment and link to a URL. You can optionally remove the formatting and create the note in plain text format.
4. Click **OK**.

To edit an attachment

1. Select an HTML file in the **Attachments** list, and click **Edit HTML**.
2. In the **Attachment Editor** dialog box, apply your edits and click **OK**.

To copy an attachment

1. Select a file in the **Attachments** list, and click **Save As**.
2. In the **Save As** dialog box, type a new name for the file.
3. Click **Save**.
4. Click **OK**.

To delete an attachment

1. Select the file you want to remove on the **Attachments** page and click **Remove**.

   The attachment disappears from the list.
2. Click **OK**.

Displaying Change Requests

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane. A list of change requests for the current folder displays in the upper pane of the project view window. The **Change Request** menu item also becomes available on the menu bar.

All change requests shown in the upper pane:

- Are attached to the folder selected from the folder tree.
- Match the filter selected from the **Filter** list.
- Match the depth specified by the **All Descendants** button.

Viewing Unread Change Requests

1. Look for change requests in bold. These are the change requests you are responsible for but have not yet reviewed.

   Note: Change requests in regular type are those that you have read or those for which you are not responsible.
2. Click on the **Responsibility** column header, then scroll down to your name. All change requests in bold are unread.

Moving Change Requests

1. Locate the change request you want to move. You can move a change request from one folder to another.

2. Click on the change request and drag it to a new folder.

Sharing Change Requests

1. `Ctrl-click` the change request and drag it to the folder that you want to share the change request.

   Note: You can also share change requests with other views and projects as long as they use the same server configuration.
2. Click **OK**.

Updating a Group of Items

The server configuration settings must allow bulk update in the workflow rules.

There are times when you may want to update several items at the same time. Items that you can edit in a group are Requirements, Change Requests, Tasks, and etc.

1. Navigate to and select the items you want to edit.
2. Right-click and choose **Bulk Update**. The dialog appropriate to the items appears.
3. Make updates to fields.
4. Click **OK**. The selected items are updated.

Default and Required Change Request Fields

The following table lists the fields on the **Change Request** dialog box, explains their uses, and indicates which fields are required.

Field	Required?	Description	Example
Status	Yes	For new change requests, set the **Status** field to `New`. The **Status** is changed to `Open` when the change request is assigned to a developer.	In this example, the status should be **New**.
Severity	Yes	Specify the seriousness of the problem. High severity items are usually associated with data loss or corruption, system crashes, etc. Low severity items are generally misspelled items and cosmetic errors.	In this example, the problem is comparatively minor (that is, if it does not cause the system to crash or lose data), so classify it as **Medium**
Priority	Yes	In most defect tracking systems, **Priority** is a multi-level choice (usually on a 1 to 5 scale). In StarTeam, however, it is a **Yes** or **No** choice. The priority of a change request is sometimes determined by the tester and sometimes by the developer. In most cases, it reflects the need to get a particular defect fixed before others. If the defect is catastrophic or prevents your team from accessing other major areas of the application, select the **Priority** field.	In this example, leave the **Priority** field cleared.
Platform	Yes	Indicate what type of operating system environment the defect occurs in. If the defect happens only on Windows 10, select Windows 10. In most cases, the defect will appear on all platforms.	In this example, set **Platform** to **All**.
External Reference	No	Specify information received from outside the company, such as a note about a defect from an outside testing service or a customer.	In this example, leave the field empty.

Field	Required?	Description	Example
		Currently this field is not used.	
Component	No	Identify the component of the product in which the defect occurs. Currently this field is not used.	In this example, leave the field empty.
Category	No	Identify a sub-component of the product. It is used with the **Component** field to identify the location in which the defect occurs. Currently this field is not used.	In this example, leave the field empty.
Synopsis	Yes	Use to give a brief summary of the problem encountered or the suggested enhancement. Consider the synopsis to be a title for the defect. **Note:** The **Synopsis** should only contain information for one defect. If the reported defect uncovers or relates to another defect, the second defect should be written up separately and referenced to the first defect in the synopsis (for example, "`CR #3109 also relates to this defect`").	For this example, a synopsis might be: "`Available fields disappear when using the Advanced Fields box.`"
Type	Yes	If the change request is a reproducible problem in the software, select **Defect**. If it is a customer request or a feature enhancement request, select **Suggestion**.	For this example, select **Defect**.
Last Build Tested	Yes	Indicate the build number of the software in which the defect was discovered or last tested. If you are writing a change request, select the build number from the application (often found in the **About** dialog). If you are verifying or regressing the change request, and the problem still exists in the current build, change this field to the build number you are currently testing.	For this example, select the most current build number.
Addressed in Build	Yes	Indicate the build in which the fix first appears. In most cases, after the engineer fixes the defect, the field will be set to **Next Build**. This field changes to the correct build when that version is actually built.	For this example, leave the field empty.
Responsibility	No	Indicate the person who should act on the defect. Depending on the position of the change request in the change request life cycle, this person could be a developer, a QA engineer, or the person who first reported the change request.	For this example, either leave this field blank, or assign it to the lead engineer on the project, who will assign it to the appropriate person.
Addressed by	Yes	This field is automatically filled with the name of the person who originally wrote up the change request. It is not editable.	NA
Description/**Steps to Reproduce**	Yes	Select the **Description Tab**. In the **Description/ Steps to Reproduce** field, enter detailed information about the defect. Specifically, the description should build on the synopsis information. The **Steps to Reproduce** information is the most important data entered in the change request	Steps to reproduce might look as follows: **1.** Click the column headers in the upper pane. **2.** Select **Show Fields**.

Field	Required?	Description	Example
		because it provides a detailed step-by-step method of reproducing the defect. The more detailed the information, the more likely the responsible developer will be able to determine the cause of the defect and fix the defect.	**3.** Click **Show Advanced Fields** check box. The check box is activated. **4.** Click Show **Advanced Fields** check box. The check box is deactivated. **5.** EXP: The standard fields appear in the **Available Fields** list. **6.** ACT: No fields appear in the **Available Fields** list.

Specifying Change Request Descriptions

You use the **Description** tab to specify detailed information about the change request including the steps to reproduce the problem.

1. Click the **Description** tab.
2. Type a detailed description of the change request in the **Description and steps to reproduce** field.

   Include the steps to reproduce the problem, or in the case of an enhancement request, a detailed description of the enhancement.
3. Optionally, type or browse for the path to a test for the change request in the **Test command** field.
4. Click **Apply**.

Specifying Change Request Solutions

You use the **Solution** tab to specify a workaround for the problem and to document how this change request was resolved.

1. Click the **Solution** tab.
2. Optionally, in the **Work around** field, type the steps you can follow to work around the problem.
3. Optionally, in the **Fix** field, type the solution to the problem. The **Fix** field is usually completed by the user who fixes the code. In this field, the application can accept a maximum of 20K characters, but your database may accept fewer characters.
4. Click **Apply**.

Modifying Custom Options for Change Requests

Your team leader may have created additional change request properties. You use the **Custom** tab to change the default properties.

To set the values for custom properties:

1. Double-click a custom property on the **Custom** tab to open the **Edit Property** dialog box.
2. Type or select a new value for the property by double-clicking on the field:

integer, text, and real fields	**Value** is a text box.
enumerated types and user IDs	**Value** is a list box.

dates and times	**Value** has a **Date** check box and a **Time** check box, each of which is followed by a date or time in the format for your locale.

Tip: To enter a blank value for a **GroupList** or **UserList** property, click on a selected row to deselect it. When the item is no longer highlighted, click **OK**.

3. Click **Apply**.

Specifying Change Request Summary Information

You use the **Synopsis** tab to define and modify the summary information about a change request. Summary information includes important criteria like status, severity, and who is currently responsible for this change request.

1. On the **Synopsis** tab, accept the default status **New** or select another status from the **Status** list.
2. Indicate the severity of the change request by selecting **High**, **Medium**, or **Low** from the **Severity** list.

   Note: The team leader usually sets the criteria for high, medium and low status.
3. If the change request needs immediate attention, select **Yes** from the **Priority** list.
4. To specify the type of change request, select **Defect** or **Suggestion** from the **Type** list.
5. Select the platform to which the change request applies from the **Platform** list.
6. Type a summary of the change request in the **Synopsis** field. The application can accept a maximum of 20K characters in this text box, but your database may accept fewer characters.
7. Select the name of the team member responsible for correcting the change request from the **Responsibility** list.
8. Click **Apply**.

Assigning Change Requests

Assigning a change request refers to assigning the status of the change request as well as who is currently responsible for the change request.

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.

   Tip: To find all change requests in a folder, click **All Descendants** on the toolbar. If desired, you can create a filter or query to find all change requests with **New** status or simply sort the **Status** column in the upper pane to find all **New** change requests.
3. Select the change request, then choose **Properties** from the **Change Request** menu or context menu. The **Change Request Properties** dialog box opens.
4. Review the settings and decide on an appropriate status. You can select **Open**, **Is Duplicate**, **As Designed**, or **Deferred**.

   - If you select **Open**, the **Responsibility** changes to the person best qualified to fix or enhance the product, as described in the change request.
   - If you select Is **Duplicate** or **As Designed**, the **Responsibility** changes to the person who submitted the change request. The assumption is that the person who submitted the change request will want to know about, verify, or perhaps challenge this change in status.
5. Click **Apply**, then click **Next** or **Previous** to review another change request.

Adding Change Request Comments

You can add comments to a change request, such as the reason for changing the change request properties.

1. Click the **Comment** tab.

   Note: The **Comment** for this revision lists any comments that were entered for the current version of the change request only. That is, each time you change the change request and type a comment, the new comment replaces the old comment when you save the change request.
2. Type your comments in the **Comment for new revision** field.

   Note: You must make a change to a property of this change request before you can type a comment.
3. Click **Apply** to save your changes.

Resolve Open Change Requests

You resolve open change requests by following the steps below. Before you start working on a change request, be aware of any processes required by your team. For example:

- Your company might require that the change request **Status** be changed to **In Progress**.
- You might be required to link open change requests to the associated file or files that need to be changed. If this is the case, when you check in a file or group of files, you can indicate the change requests that are being fixed by the files. Doing this saves the time it would take to change the status of each change request.

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.

   Tip: To find all change requests in a folder, click **All Descendants** on the toolbar. If desired, you can create a filter or query to find all change requests with **New** status or simply sort the **Status** column in the upper pane to find all **New** change requests.
3. Double-click the change request. The **Change Request <number, revision #>** dialog box opens.
4. Change the **Status** of the change request to one of the resolved statuses: **Fixed**, **Documented**, or **Cannot Reproduce**. Alternatively, you might use **Is Duplicate** or **As Designed**, if either of these is appropriate.

   When you select a resolved status, StarTeam automatically makes the following changes to the change request:

   - Places the name of the person who submitted the change request in the **Responsibility** field. The assumption is that the person who submitted the change request will want to know about, verify, or perhaps challenge the change in status.
   - Changes the setting for the **Addressed in build** field to **Next Build** (if the status has changed to **Fixed** or **Documented**). When the next build label is created, **Next Build** changes to the name of the build label. The assumption is that the person who verifies that the change request has been implemented should test the correct build of the product.
5. If you choose **Fixed** or **Documented** as the new status, select the **Solution** tab and type the appropriate information in the **Work Around** and/or **Fix** text boxes.

   Often a change request suggests one or more fixes for a problem, and none of these suggestions are implemented. To avoid confusion, the fix that is implemented must be described in precise detail. Testers and writers rely heavily on this information.
6. Click **OK**.

Tip: Although the application makes these automatic changes immediately, you can change the **Responsibility** or **Addressed in build** setting before you click **OK** (or **Apply**, if appropriate). In this way, you can bypass the automatic workflow and route the change request as your team requires.

Verifying Resolved Change Requests

You verify resolved change requests by following the steps below. If you determine that a change request is not really resolved, you can reopen it.

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.

   Tip: To find all change requests in a folder, click **All Descendants** on the toolbar. If desired, you can create a filter or query to find all change requests with **New** status or simply sort the **Status** column in the upper pane to find all **New** change requests.
3. Double-click the change request. The **Change Request <number, revision #>** dialog box opens.
4. Change the status to **Open** or **Verified**. StarTeam has the following verified statuses:
   - `Verified As Designed`
   - `Verified Cannot Reproduce`
   - `Verified Documented`
   - `Verified Fixed`
   - `Verified Is Duplicate`
5. If you change the status to **Open**, type the word `Reopen` and the date in the **Synopsis** field. Otherwise, the team member who resolved the change request may think that he or she forgot to mark it resolved and, without investigating further, mark it resolved a second time.

   When you re-open a change request, StarTeam automatically does the following:
   - Places the name of the person who resolved the change request in the **Responsibility** field. The assumption is that the person who resolved the change request the first time should be the person to continue working on it.
   - Blanks out the setting for the **Addressed in build** field. The assumption is that the change request has not been resolved and, therefore, has not been addressed in any build.
6. Do one of the following:
   - Click **Apply**, then click the **Next** or **Previous** button to verify another change request.
   - Click **OK**.

Closing Verified Change Requests

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.

   Tip: To find all change requests in a folder, click **All Descendants** on the toolbar. If desired, you can create a filter or query to find all change requests with **New** status or simply sort the **Status** column in the upper pane to find all **New** change requests.
3. Double-click the change request. The **Change Request <number, revision #>** dialog box opens.
4. Change the status to **Closed**. StarTeam has the following closed statuses:
   - `Closed (As Designed)`
   - `Closed (Cannot Reproduce)`
   - `Closed (Deferred)`
   - `Closed (Documented)`
   - `Closed (Fixed)`
   - `Closed (Is Duplicate)`
5. Do one of the following:

- Click **Apply**, then click the **Next** or **Previous** button to close another change request.
- Click **OK**.

Customizing Change Request Filters

After you have sorted, grouped, selected columns, applied queries to the change requests component in the upper pane, you can save the arrangement of change request data that appears in the upper pane as a filter. You can later apply the filter to any change request data to view the data using the same arrangement.

1. Right-click a column header and select **Save Current Settings**. The **Save Current Settings** dialog appears.
2. Type a filter name in the **Filter Name** field.
3. Select or clear the **Public** check box depending on whether this filter is to be used by all or only on your workstation.
4. Click **OK**. The filter name will appear in the **Filter** list.

Selecting Change Requests Using a Query

You can use a simple or complex query to limit the change requests displayed to those that fit specific criteria.

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.
3. Right-click a column header and choose **Queries**.
4. Choose an existing query in the **Queries** dialog box.
5. Click **Select**.
6. Follow the steps in *Creating Queries* to define a new query.

Reviewing Linked Change Requests

If you are checking in a file that has one or more linked change requests, you should also review all change requests associated with the file.

1. Select the file in the upper pane that is linked to the change request.
2. Choose **File** > **Check In** . The **Check In** dialog box opens.
3. Click **Advanced** to open the **Advanced Options** dialog box.
4. Click **Show Change Requests**.

   The **Advanced Options** dialog box expands at the bottom and displays the list **Change Requests Linked In This View**.

   Note: No change request appears in the list more than once, even if it is linked to several of the files you are checking in. When a change request is linked to more than one file, the list displays the name of only one of the files.
5. Optionally, double-click a change request to review or edit its properties.
6. Optionally, check **Marked Selected Change Requests As Fixed**.

   If you check this option, StarTeam will mark the selected, but unresolved, change request **Fixed** as part of the check-in process.

Sorting and Grouping Change Requests

You can sort change requests by the data in a particular column or group the change requests.

1. Click on a column header on the **Change Requests** tab to sort the change requests by the data in that column.

   Note: If you want to sort or group the change requests first by the data in one column and then by the data in another column, see the following steps. You can sort or group the change requests in up to four levels of groupings.
2. To sort or group the change requests in multiple levels, right-click the column headers and select **Sort and Group**. The **Sort and Group** dialog box opens.
3. From the **First By** list, select a column title.
4. Optionally, group the change requests by the data in this column, select the **Group By** check box.

   If you select the **Group By** check box, the change requests are grouped together in nested lists and you must drill down to view the change requests in each group. If you do not select the **Group By** check box, the change requests are all displayed on the **Change Requests** tab, sorted by your choices in **Sort and Group** dialog box.

   Note: By default, the column data is sorted or grouped based on the internal key or order. You can use the **Sort Options** button and choose to sort or group the data based on the text and optionally, case sensitivity.
5. Repeat steps three and four to define up to four levels of sort orders or groupings.
6. Click **OK**.

Showing Fields in a Change Request

You can select which fields are displayed for a change request.

1. Right-click a column headers and select **Show Fields**.

   The **Show Fields** dialog box opens.
2. Make sure the **CR Number**, **Entered By**, and any other appropriate fields are displayed in the **Show these fields in this order** list.
3. Click **OK**.

Customizing Change Request Reports

You can create a number of reports using the change request report features. This topic demonstrates an example of creating a report of change requests fixed during a certain time period.

1. Select a folder from the folder tree.
2. Click the **Change Requests** tab in the upper pane.
3. Click **Change Request** > **All Descendants** .
4. On the **Change Request** tab, display the **CR Number**, **Status**, **Modified Time**, and **Modified By** fields.
5. Define the query that includes these fields and specifies a date range.

   Specify a beginning **Modified Time**, and if the end date is not the current date, use an `AND` operator and specify an ending **Modified Time**.
6. Sort and group the change requests, selecting **Status** from the **First By** drop-down list box and checking **Group By**.
7. Select the **Status: Fixed** group.
8. Choose **Change Request** > **Reports** to create a report showing the fixed change requests.

Change Request Properties

This topic presents the change request properties and their descriptions as displayed in the **Change Request Properties** dialog box. The **Change Request Properties** dialog box contains the following tabbed pages of properties:

Synopsis

The following properties are on the **Synopsis** page.

Status	Displays the status of the change request.
Priority	Displays the priority level of the change request. Many people use repository customization to extend this field to include other values because Boolean values in the application are treated as enumerated types. For example, No is 0 and Yes is 1. An administrator might change `No` to `Not A Priority`, `Yes` to `Priority 1`, and add `Priorities 2` through `10`.
Type	Displays the type of change request, a **Defect** or a **Suggestion**.
Severity	Indicates the severity of the change request: **Low**, **Medium**, or **High**.
Platform	Indicates which operating system platform the to which the change request applies.
Last Build Tested	Displays the build label selected by a user to represent the last build in which a change request was tested.
External Reference	Indicates the customer or other outside source who provided the data for this change request.
Addressed In Build	Indicates the next build label created and applied to the view after the resolution to a change request occurs.
Component	Displays the component in which the defect occurs. It is often used with the **Category** property to narrow that identification to a sub-component.
Category	Displays the name of the sub-component in which the defect occurs. It is usually used in combination with the **Component** property.
Synopsis	Displays a brief description of the change request.
Responsibility	Displays the name of the person currently responsible for the change request.
Entered By	Displays the name of the person who entered the change request.

Description

The following properties are on the **Description** page. This page also contains a **Browse** button for locating the command to test and a **Run** button for running the test.

Description And Steps To Reproduce	Displays a detailed description of the change request.
Test Command	Displays the command to use to test the solution for the change request.

Solution

The following properties are on the **Solution** page.

Work Around	Explains the solution to the change request other than the fix.
Fix	Displays the solution to the problem addressed by the change request.

Custom

You can create custom properties for an item which will display in the item **Properties** dialog box.

The following properties are on the **Custom** page.

Property	Displays each custom property name.
Value	Displays the values for each custom property. Double-click the property name to edit the value.

Attachments

The **Attachments** page contains a list of all the files attached to the current change request.

Comment

The following properties are on the **Comment** page.

Comment For This Revision	Displays the reason for the changes to the current revision.
Comment For New Revision	Displays the reason for the changes to the new revision.

Change Request Fields

This section lists all the change request fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or StarTeam Notification Agent . Reports can use any field name.

Addressed By

Values: list of users, `<None>`.

Internal Identifier: `AddressedBy`.

Indicates the user who resolved a change request (resolved statuses are `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, and `Is Duplicate`).

Addressed In

Values: list of view labels, `<None>`.

Internal Identifier: `AddressedIn`.

Indicates the next build label created and applied to the view after the resolution to a change request occurs.

Addressed In View

Values: list of views, `<None>`.

Internal Identifier: `AddressedInView`

Indicates in what view the change request has been resolved. This is important for shared, and perhaps moved, change requests.

Attachment Count

Values: `number`.

Internal Identifier: `AttachmentCount`.

The number of files attached to an item.

Attachment IDs (Advanced)

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. For example: `[00 00 00 00 02 00 00 00]` indicates two specific attachments.

Internal Identifier: `AttachmentCount`.

Cannot be used in queries. The ID numbers assigned to attachments. For example, the first attachment within a project is `00 00 00 00`.

Attachment Names

Values: `text` containing a series of file names separated by spaces.

Internal Identifier: `AttachmentNames`.

The names of the files attached to an item.

Branch On Change (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `BranchOnChange`.

Indicates whether the item will branch when it changes.

The value is `No` if the item's behavior is not set to **Branch On Change**. Reasons for this may be:

- The item is in the root or a reference view and the **Branch On Change** feature is disabled.
- The item is in a branching view but has already branched as a result of a change, which, in turn, results in the **Branch On Change** feature becoming disabled.
- The item is in a branching view, but its behavior currently does not permit it to branch on change. This means that modifications are checked into the parent view.

Note: If the value is `No`, the value of the **Branch State** explains the `No`.

Branch State (Advanced)

Values: `Branched`, `Not Branched`, `Root`.

Internal Identifier: `BranchState`.

Indicates whether an item has branched in the child view, is still unbranched (and therefore is part of the parent view), or was created in the view in which it resides.

The values `Branched` and `Not Branched` apply to items in branching views. The value `Root` applies to items created in the view in which the item currently resides.

If the view is a reference view, it reflects the state of the item in the reference view’s parent.

Category

Values: `text`.

Internal Identifier: `Category`.

Text identifying the sub-component in which the defect occurs. It is usually used in combination with the **Component** field.

Closed On

Values: `date/atime`.

Internal Identifier: `ClosedOn`.

The date and time at which a change request was closed.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those 2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Component

Values: `text`.

Internal Identifier: `Component`.

Text identifying the component in which the defect occurs. It is often used with the **Category** field to narrow that identification to a sub-component.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created or the time at which the label associated with the promotion state was created.

CR Number

Values: `number`.

Internal Identifier: `ChangeNumber`.

The number assigned to a change request. For example, if the **Object ID** is `0`, the change request number is `1`.

Created By

Values: list of users, `<None>`.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Deleted By

Values: list of users, `<None>`.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Description

Values: `text`.

Internal Identifier: `Description`.

The description provided for an item at the time it was added to the view, including any later edits to it.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

Entered By

Values: list of users, `<None>`.

Internal Identifier: `EnteredBy`.

The name of the user who created this change request.

Entered On

Values: `date/time`.

Internal Identifier: `EnteredOn`.

The time at which this change request was created.

External Reference

Values: `text`

Internal Identifier: `ExternalReference`.

Text usually used to indicate a customer or other outside source who provided the data for this change request.

Fix

Values: `text`.

Internal Identifier: `Fix`.

The text in the **Fix** field.

Flag

Values: `No`, `Yes`.

Internal Identifier: `Flag`.

Marks or bookmarks files in the upper pane on your workstation. This is a client-calculated field.

Flag User List (Advanced)

Values: text displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `FlagUserList`.

Can be used in queries. Identifies users who have set flags on a given item.

Folder

Values: `text`.

Internal Identifier: `Folder`.

The name of the folder that stores the item. This is a client-calculated field.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

Item Deleted By

Values: list of users, `None`.

Internal Identifier: `ItemDeletedUserID`.

The name of the user who deleted this item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Item Deleted Time

Values: `date/time`.

Internal Identifier: `ItemDeltedTime`.

The time at which the item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Last Build Tested

Values: list of view labels, `<None>`.

Internal Identifier: `LastBuildTested`.

The build label selected by a user to represent the last build in which a change request was tested.

Locked By

Values: list of users, `<None>`.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

Modified By

Values: list of users, `<None>`.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

My Lock

Values: `Exclusively Locked By Me`, `Non-exclusively Locked By Me`, `Not Locked By Me`.

Internal Identifier: `MyLock`.

Indicates whether the current user has the item locked and, if so, whether that lock is exclusive or not. This is a client-calculated field.

New Revision Comment (Advanced)

Values: `text`.

Internal Identifier: `NewRevisionComment`.

Internal use only. The client uses this value during the item update process. The field always appears empty if added to the upper pane. This is a client-calculated field.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Parent Branch Revision (Advanced)

Values: `number`.

Internal Identifier: `ParentRevision`.

The last digit in the branch revision number before an item branched. For example, if this number is `7`, the branch revision was `1.7` at the time the item branched (becoming `1.7.1.0`, as seen in the item's history). This number is `-1` if an item was not inherited from the parent view.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Parent Revision (Advanced)

Values: `number`.

Internal Identifier: `PathRevision`.

The revision number at which an item branched. For example, if this number is `8`, this item's revision number in the parent view was `8` at the time the item branched. The history should show that revision `9` in the first revision in the current view. This number is `0` if this item was not inherited from the parent view.

Platform

Values: `All`, `MacOS`, `Other`, `Unix`, `Windows 2000`, `Windows 95`, `Windows 98`, `Windows NT`, `Windows XP`.

Internal Identifier: `Platform`

The value of the **Platform** field.

Priority

Values: `No`, `Yes`.

Internal Identifier: `Priority`.

The value of the **Priority** field. Many people use repository customization to extend this field to include other values because booleans in the application are treated as enumerated types. For example, `No` is `0` and `Yes` is `1`. An administrator might change `No` to `Not A Priority`, `Yes` to `Priority 1`, and add `Priority 2` through `Priority 10`.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item's configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Read Status

Values: `Read`, `Unread`.

Internal Identifier: `ReadStatus`.

Indicates whether an item is considered read or not read. This is a client-calculated field.

Read Status User List

Values: `text` displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `ReadStatusUserList`.

Can be used in queries. Identifies users for whom a given item's status is **Unread**.

Resolved On

Values: `date/time`.

Internal Identifier: `ResolvedOn`.

The time at which a change request was resolved. The resolution can be: **Cannot Reproduce**, **As Designed**, **Fixed**, **Documented**, or **Is Duplicate**.

Responsibility

Values: list of users, `<None>`.

Internal Identifier: `Responsibility`.

The name of the user who is currently responsible for a change request.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Root Object ID (Advanced)

Values: `number`.

Internal Identifier: `RootObjectID`.

The object ID of the oldest ancestor of an item. For example, if an item was not inherited from a parent view, the root object ID is the same as its object ID. If it was inherited from a parent view, the root object ID is the Parent ID, or the item's Parent ID.

Severity

Values: `High`, `Low`, `Medium`.

Internal Identifier: `Severity`.

The value of the **Severity** field.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item's properties or contents. Additional text is stored in the **Comment** field.

Status

Values: `New, Open, In Progress, Deferred, Cannot Reproduce, As Designed, Fixed, Documented, Is Duplicate, Verified Deferred, Verified Cannot Reproduce, Verified As Designed, Verified Fixed, Verified Documented, Verified Is Duplicate, Closed`

`Deferred`, `Closed Cannot Reproduce`, `Closed As Designed`, `Closed Fixed`, `Closed Documented`, `Closed Is Duplicate`.

Internal Identifier: `Status`.

The value of the **Status** field.

Synopsis

Values: `text`.

Internal Identifier: `Synopsis`.

The value of the **Synopsis** field.

Test Command

Values: `text`.

Internal Identifier: `TestCommand`.

The text in the **Test Command** field.

Type

Values: `Defect`, `Suggestion`.

Internal Identifier: `Type`.

The value of the **Type** field.

Verified On

Values: `date/time`.

Internal Identifier: `VerifiedOn`.

The time at which a change request was verified. The resolution can be **Verified Cannot Reproduce**, **Verified As Designed**, **Verified Fixed**, **Verified Documented**, or **Verified Is Duplicate**.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is `2`.

View

Values: list of views, `<None>`.

Internal Identifier: `ViewID`.

The name of the view in which the item last branched. For example, if an item is inherited from a parent view but is branched in a child view, the value of this field in the child view changes from the name of the parent view to the name of the child view for the revision that branched and subsequent revisions in the child view.

Work Around

Values: `text`.

Internal Identifier: `WorkAround`.

The text in the **Work Around** field.

Requirements

Requirements are supported for the Enterprise Advantage license and display in the **Requirement** tab of the upper pane in the clients. With the requirement component, you can create requirements within the application and show the dependencies among them. For example, if one requirement must be fulfilled before a second requirement can be fulfilled, the first can be made a child of the second. If your company enforces process rules, the requirements you establish can also be used to drive the development process.

Administrators and other authorized users can publish requirements from Caliber to StarTeam using Publisher to StarTeam, which is delivered with Caliber.

Requirement Characteristics

The requirements in the upper pane have the following characteristics:

- They are attached to the folder selected from the folder hierarchy.
- They match the filter selected from the **Filter** list.
- They match the depth specified by **All Descendants**.

  Tip: You can click the button on the toolbar or click **Requirements** > **All Descendants**.

  Note: You can click the **All descendants** button on the **Requirements** view toolbar.

Note: Icons display to the left of a requirement in the upper pane to indicate its status and whether you have read the latest revision.

How Requirements Can Help

By using a requirements-driven development processes, companies can prevent consuming, costly misunderstandings and shorten time to market. To accomplish this, you can use the StarTeam built-in requirement component as your basic tool, or publish complex requirements to StarTeam from Caliber. Using requirements enables business analysts, managers, developers, QA staff, and others to:

- Organize business, user, and functional requirements in a hierarchical format.
- Indicate the dependencies among requirements.
- See all layers of requirements at all times.
- Prioritize requirements by importance.
- Identify the impact of changes to requirements.
- Use requirements to estimate work.
- Identify the person creating the requirement.
- Notify those who will be responsible for fulfilling the requirements.
- Track the requirement life-cycle from submitted to completed or rejected.
- Provide requirements with a context by linking them to files, change requests, and topics.

Creating Requirements

Creating a hierarchy of requirements allows you to organize a project efficiently and work toward agreed-upon goals.

1. Click the **Requirement** tab on the upper pane.
2. Do one of the following:
   - Choose **Requirement** > **New** to create a new requirement that is not the child of an existing requirement.
   - Choose **Requirement** > **New Child Requirement** to create a new requirement that is the child of an existing requirement.

   The **New Requirement** dialog box appears.
3. Click the **Requirement** tab of the **New Requirement** dialog box and do the following:
   a) Type a name for the requirement.
   b) Select an owner (for example, the person ultimately responsible for the fulfillment of the requirement) from the **Owner** list.

c) Optionally, provide an external source or reference for the requirement in the **External reference** field. If you publish requirements from CaliberRM to StarTeam, this field displays the CaliberRM identification for this requirement.

d) Type the initial description of this requirement in the **Description** field. This description is usually revised over time to eliminate ambiguities.

4. Click the **Responsibility** tab and list the team members responsible for this requirement. If notification is enabled, these people will be notified about changes made to any field in the requirement.

   a) Click **Add** to display the **Select Responsible Users** dialog box.

   b) Double-click the name of each person to be added to the list. When you double-click the name, it moves from the **Users** list to the **Responsible Users** list.

   c) Add the remaining responsible users to the **Responsible Users** list box and click **OK**.

5. Use the **Estimate** tab to indicate the best-case and worst-case times for fulfilling this requirement. The entries are usually in staff days.

   - Type the number of units (usually days) estimated for the fulfillment of this requirement in the **Expected effort** text box.
   - Type the number of units (usually days) estimated for the worst-case fulfillment of this requirement in the **High effort** text box.
   - Type the number of units (usually days) estimated for the best-case fulfillment of this requirement in the **Low effort** text box.
   - Add any appropriate notes in the **Notes** text box.

6. Use the **Custom** tab to provide values for any custom requirement properties that your team leader or company may have created. Double-click a custom property on the **Custom** tab to open the **Edit Property** dialog box.

integer, text, and real fields	**Value** is a text box.
enumerated types and user IDs	**Value** is a list box.
dates and times	**Value** has a **Date** check box and a **Time** check box, each of which is followed by a date or time in the format for your locale.

   Tip: To enter a blank value for a **GroupList** or **UserList** property, click on a selected row to deselect it. When the item is no longer highlighted, click **OK**.

   Click **Apply** to save your changes.

7. Use the **Comment** tab to explain why the requirement is being created or revised. Enter your reasons in the **Comment for new revision** text box.

8. Click **OK**.

Working with Attachments

You can attach image files, HTML/plain text formatted notes, and other file types to a component using the **Attachments** tab on the **Properties** dialog box of the selected component. The HTML/plain text attachment is similar to a note in which you can easily format and align text without having to use an external editor.

Note: Attachments can only be used with Change Request, Requirement, Task, and Topic components.

1. Select a folder from the folder hierarchy and click the component tab in the upper pane (for example, **Change Request**) .
2. Open a component item that you want to add an attachment to, delete an attachment, or edit an existing attachment.
3. Click the **Attachments** tab.

To add an attachment

1. Click **Add**. The **Open** dialog box displays.
2. Select the file you want to attach to the component and click **Open**. To select multiple items, use the keyboard functions **Shift+Click** or **Ctrl+Click**. The selected item appears in the **Attachments** list on the **Attachments** page of the **Properties** dialog box.
3. Click **OK**.

To add an HTML or plain text attachment

1. On the **Attachments** tab, click **Add HTML**. The **Attachment Editor** dialog box displays.
2. In **Name**, type a descriptive file name or title to identify the attachment.
3. In the text box, enter descriptive content for the body of the attachment and apply the formatting of your choice. You can also insert an image in the attachment and link to a URL. You can optionally remove the formatting and create the note in plain text format.
4. Click **OK**.

To edit an attachment

1. Select an HTML file in the **Attachments** list, and click **Edit HTML**.
2. In the **Attachment Editor** dialog box, apply your edits and click **OK**.

To copy an attachment

1. Select a file in the **Attachments** list, and click **Save As**.
2. In the **Save As** dialog box, type a new name for the file.
3. Click **Save**.
4. Click **OK**.

To delete an attachment

1. Select the file you want to remove on the **Attachments** page and click **Remove**.

   The attachment disappears from the list.
2. Click **OK**.

Marking Item Threads Read or Unread

You can display requirement, task, and topic components in either a list display or hierarchical format. StarTeam provides a default view for requirement, task, and topic components in a hierarchical structure. When you select to view the hierarchical structure, each new item becomes the root of a tree. Its branches are child requirements, subtasks to the parent task, or responses to the topic. Children of children requirements, subtasks of subtasks, and responses to responses form additional branches. When an item thread is unread, the textual information about the item thread displays in bold text in upper pane. When an item thread is read, the item thread displays in normal text in the upper pane.

The main menu or context menu commands enable you to mark item threads as read or unread to better track these components.

1. Select a topic thread in the upper pane.

   Tip: To select multiple threads, display the information in list format by clicking the **List Display** button in the toolbar.
2. Do one of the following:
   - Choose **Mark Thread as Read** to remove the bold format for the item thread.
   - Choose **Mark Thread as Unread** to add bold format for the item thread.

Updating a Group of Items

The server configuration settings must allow bulk update in the workflow rules.

There are times when you may want to update several items at the same time. Items that you can edit in a group are Requirements, Change Requests, Tasks, and etc.

1. Navigate to and select the items you want to edit.
2. Right-click and choose **Bulk Update**. The dialog appropriate to the items appears.
3. Make updates to fields.
4. Click **OK**. The selected items are updated.

Requirement Properties

This topic presents the requirement properties and their descriptions as displayed in the **Requirement Properties** dialog box. The **Requirement Properties** dialog box contains the following tabbed pages of properties.

Requirement

The following properties are on the **Requirement** page.

Name	Displays the requirement name.
Created By	Displays the name of person who created the first revision of the requirement in the view.
Created On	Displays the date on which first revision of the requirement was created.
Attachments	Indicates the number of files attached to the requirement.
Modified By	Displays the name of the last person who last modified the requirement.
Modified On	Displays the date on which the requirement was last modified.
Type	Displays the requirement type.
Owner	Displays the name of person ultimately responsible for the fulfillment of the requirement.
Status	Displays the current status of the requirement. This indicates the progress from submitted to rejected or completed. Note: The status **ReadyForCCP** means the requirement is ready for review by the Change Control Board.
External Reference	External source or reference for this requirement. This usually is the name of an external customer who asked for the requirement. If you are publishing requirements from CaliberRM to StarTeam, this property displays its identification for this requirement.
Description	Provides a description of the requirement, usually revised over time to eliminate ambiguities.

Responsibility

The **Responsibility** page lists the people responsible for completion of the requirement. You can add or remove people from the list.

These people will be notified of changes to the requirement if notification is enabled.

Ambiguity Review

The following properties are on the **Ambiguity Review** page. Reviewers will use the **Ambiguity Review** page to locate ambiguities in the initial description and revise that description.

Number Of Ambiguities Found	Indicates the number of ambiguities reviewers have found in the initial description of the requirement.

Revised Description Provides a new, revised description because of ambiguities found in the original description or for other reasons.

Comments Provides comments stating what the ambiguities are in the original requirement and why you have made the changes to the description.

Estimate

The following properties are on the **Estimate** page.

Expected Effort Indicates the expected case estimate for how long it will take to implement the requirement fully. If you are publishing requirements from CaliberRM to StarTeam, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Low Effort** and the **High Effort** fields, and should be used consistently for all requirements.

High Effort Indicates the worst case estimate for how long it will take to implement the requirement fully. If you are publishing requirements from CaliberRM to StarTeam, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Low Effort** and the **Expected Effort** properties, and should be used consistently for all requirements.

Low Effort Indicates the best case estimate for how long it will take to implement the requirement fully. If you are importing requirements from CaliberRM, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Expected Effort** and the **High Effort** fields, and should be used consistently for all requirements.

Custom

You can create custom properties for an item which will display in the item **Properties** dialog box.

The following properties are on the **Custom** page.

Property Displays each custom property name.

Value Displays the values for each custom property. Double-click the property name to edit the value.

Attachments

The **Attachments** page contains a list of all the files attached to the current requirement.

Comment

The following properties are on the **Comment** page.

Comment For This Revision Displays the reason for the changes to the current revision.

Comment For New Revision Displays the reason for the changes to the new revision.

Requirement Fields

This section lists all the requirement fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or StarTeam Notification Agent . Reports can use any field name.

Am I Responsible? Values: `No`, `Yes`.

Internal Identifier: `AmIResponsible`.

Indicates whether the logged-on user is responsible for a requirement. This is a client-calculated field.

Ambiguities Found

Values: `number`.

Internal Identifier: `AmbiguitiesFound`.

Indicates the number of ambiguities found in the requirement.

Attachment Count

Values: `number`.

Internal Identifier: `AttachmentCount`.

The number of files attached to an item.

Attachment IDs (Advanced)

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. For example: `[00 00 00 00 02 00 00 00]` indicates two specific attachments.

Internal Identifier: `AttachmentCount`.

Cannot be used in queries. The ID numbers assigned to attachments. For example, the first attachment within a project is `00 00 00 00`.

Attachment Names

Values: `text` containing a series of file names separated by spaces.

Internal Identifier: `AttachmentNames`.

The names of the files attached to an item.

Children Count

Values: `number`.

Internal Identifier: `ChildrenCount`.

The number of items that are children of this item. This is a client-calculated field.

ChildType

Values: `Child Requirement`, `Requirement`.

Internal Identifier: `ChildType`.

Indicates whether the requirement is the root of a requirement tree or a child of another requirement. This is a client-calculated field.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those 2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Comments

Values: `text`.

Internal Identifier: `Comments`

Provides comments about the revised description created because of ambiguities found in the original description or for other reasons.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created or the time at which the label associated with the promotion state was created.

Created By

Values: list of users, `<None>`.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Deleted By

Values: list of users, `<None>`.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Description

Values: `text`.

Internal Identifier: `Description`.

The description provided for an item at the time it was added to the view, including any later edits to it.

Disabled

Values: `No`, `Yes`.

Internal Identifier: `Disabled`.

Indicates whether the requirement is disabled.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

Expected Effort

Values: `number`.

Internal Identifier: `ExpectedEffort`.

Indicates the expected case estimate for how long it will take to implement the requirement fully. If you are publishing requirements from CaliberRM to StarTeam, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Low Effort** and the **High Effort** fields, and should be used consistently for all requirements.

External Reference

Values: `text`.

Internal Identifier: `ExternalReference`.

Usually provides the name of an external customer who asked for this requirement.

Flag

Values: `No`, `Yes`.

Internal Identifier: `Flag`.

Marks or bookmarks files in the upper pane on your workstation. This is a client-calculated field.

Flag User List (Advanced)

Values: text displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `FlagUserList`.

Can be used in queries. Identifies users who have set flags on a given item.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

High Effort

Values: `number`.

Internal Identifier: `HighEffort`.

Indicates the worst case estimate for how long it will take to implement the requirement fully. If you are publishing requirements from CaliberRM to StarTeam, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Low Effort** and the **Expected Effort** fields, and should be used consistently for all requirements.

Item Deleted By

Values: list of users, `None`.

Internal Identifier: `ItemDeletedUserID`.

The name of the user who deleted this item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Item Deleted Time

Values: `date/time`.

Internal Identifier: `ItemDeltedTime`.

The time at which the item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Locked By

Values: list of users, `<None>`.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

Low Effort

Values: `number`.

Internal Identifier: `LowEffort`.

Indicates the best case estimate for how long it will take to implement the requirement fully. If you are publishing requirements from CaliberRM to StarTeam, these fields will already be filled with data based on a specific unit, such as hours or days. Otherwise, the units are arbitrary, but should be the same for the **Expected Effort** and the **High Effort** fields, and should be used consistently for all requirements.

Modified By

Values: list of users, `<None>`.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

My Lock

Values: `Exclusively Locked By Me`, `Non-exclusively Locked By Me`, `Not Locked By Me`.

Internal Identifier: `MyLock`.

Indicates whether the current user has the item locked and, if so, whether that lock is exclusive or not. This is a client-calculated field.

Name

Values: `text`.

Internal Identifier: `Name`.

Displays the name of the item.

New Revision Comment (Advanced)

Values: `text`.

Internal Identifier: `NewRevisionComment`.

Internal use only. The client uses this value during the item update process. The field always appears empty if added to the upper pane. This is a client-calculated field.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Notes

Values: `text`.

Internal Identifier: `Notes`.

Text comments on the effort levels for this item.

Number

Values: `number`.

Internal Identifier: `RequirementNumber`.

Number identifying the requirement. For example, if the Object ID is 0, the requirement number is 1.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Owner

Values: list of users, <`None`>.

Internal Identifier: `Owner`.

Indicates who is ultimately responsible for this requirement.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Priority

Values: `Desirable`, `Essential`, `Unassigned`, `Useful`.

Internal Identifier: `Priority`.

The value of the **Priority** field. You can use repository customization to change the names of these values or include other values.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item’s configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Read Status

Values: `Read`, `Unread`.

Internal Identifier: `ReadStatus`.

Indicates whether an item is considered read or not read. This is a client-calculated field.

Read Status User List

Values: `text` displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `ReadStatusUserList`.

Can be used in queries. Identifies users for whom a given item’s status is **Unread**.

Responsible Count

Values: `number`.

Internal Identifier: `ResponsibleCount`.

The number of users who are responsible for a requirement.

Responsible IDs

Values: `byte array`, displayed as a bracketed series of numbers in hex format. For example, `[14 00 00 00]` indicates a specific user.

Internal Identifier: `ResponsibleIDs`.

Can not be used in queries. The ID numbers assigned to the users who are responsible for the requirement.

Responsible Names

Values: `text` containing a series of user names separated by spaces.

Internal Identifier: `ResponsibleNames`.

The names of the users responsible for this requirement.

Reviewed By

Values: `byte array`.

Internal Identifier: `ReviewedByIDs`.

Can not be used in queries. Should not be used at all.

Revised Description

Values: `text`.

Internal Identifier: `RevisedDescription`.

Provides a new, revised description because of ambiguities found in the original description or for other reasons.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item’s properties or contents. Additional text is stored in the **Comment** field.

Status

Values: `Accepted, Approved, Complete, Deferred, Draft, Pending, ReadyForCCB, Rejected, Review, Submitted`.

Internal Identifier: `Status`.

Indicates the status of this requirement.

Type

Values: `Business Requirement, Business Specification, Hardware Requirement, Hardware Specification, Human Resources, Information Technology, Software Requirement, Software Specification`.

Internal Identifier: `Type`.

Indicates the type of requirement.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is 2.

Tasks

The *task* component allows the creation of task lists and work assignments. As a standalone, the task component is very useful for managing a project. It allows team members to indicate who should do what and when, see current task status, estimate hours required to complete a task, record hours spent completing the task, and compare estimated to actual times. Because the application contains both a version control system and a change request system, it also allows tasks to be linked to the files and product defects or suggestions with which they are associated.

The task component can be used independently or interoperate with data from Microsoft Project. It can display tasks in a tree format, which clearly shows the relationship between tasks and subtasks, or in a list format, which allows tasks to be sorted, grouped, or queried, or specific fields to be selected for display. To improve efficiency, each task displays icons that identify its status, priority, milestone, and need for attention. For information about interoperating with Microsoft Project, see the *StarTeam Microsoft Project Integration User’s Guide*.

With the StarTeam task component, you can create an individual task or a summary task that has a set of subtasks. It is recommended that you plan tasks before entering them because:

- A task that has even one subtask cannot have work records added to it, although work records can be added to subtasks. The application assumes that the name of the task indicates a goal, perhaps a milestone, that will be reached when the subtasks are completed.
- After a work record has been added to a task, you cannot create subtasks for it.

Note: Regardless of whether work records can be added to a task, you should assign the responsibility for its completion to a specific team member. If work records can be added to a task, you should also estimate how long the task should take.

Creating Tasks

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Click **Task** > **New**. The **New Task** dialog box opens.
3. On the **Task** page of the **New Task** dialog box, type the **Name** of the task. You can use up to 255 characters for the task name.

   Note: Although the **Responsibility** field names the person primarily responsible for the completion of a task, additional people can be designated on the **Resources** tab.
4. Optionally, check **Milestone** to indicate that the task should be treated as a milestone task. You can display the **Milestone** column in the task list and sort for the tasks that have been designated as milestones.
5. Select the current status of the task from the **Status** list.

Pending	Waiting for completion of a predecessor task.
Ready To Start	Work can be started on the task.
In Progress	Work has been entered for the task.
Finished	Work is finished on the task.

Closed Task is completed and closed.

Hold Work temporarily stopped on the task, usually to wait for completion of another task.

6. Choose a **Priority** level from the list. The priorities are identical to those used in Microsoft Project.

   Note: Do not use the priority **Do Not Level**. This priority is a Microsoft Project-specific term.

7. Type the **Duration** which is the number of hours the task will take to complete. This field is disabled if the task contains sub-tasks since the duration of a task is dependent upon the duration of its sub-tasks.
8. Type the **Percent Complete** which is the percentage of work already completed for this task. This field may range from 0-100. The default is 0 for new tasks.
9. Optionally, check **Needs Attention** to notify team leaders or task reviewers that this task requires attention. Explain why this task needs attention in the text box. Team leaders can add the **Needs Attention** column to their task list and sort for items with this designation.
10. Click **OK**. This creates a new task which will serve as the root of a task tree in the **Task** pane.

Working with Attachments

You can attach image files, HTML/plain text formatted notes, and other file types to a component using the **Attachments** tab on the **Properties** dialog box of the selected component. The HTML/plain text attachment is similar to a note in which you can easily format and align text without having to use an external editor.

Note: Attachments can only be used with Change Request, Requirement, Task, and Topic components.

1. Select a folder from the folder hierarchy and click the component tab in the upper pane (for example, **Change Request**) .
2. Open a component item that you want to add an attachment to, delete an attachment, or edit an existing attachment.
3. Click the **Attachments** tab.

To add an attachment

1. Click **Add**. The **Open** dialog box displays.
2. Select the file you want to attach to the component and click **Open**. To select multiple items, use the keyboard functions **Shift+Click** or **Ctrl+Click**. The selected item appears in the **Attachments** list on the **Attachments** page of the **Properties** dialog box.
3. Click **OK**.

To add an HTML or plain text attachment

1. On the **Attachments** tab, click **Add HTML**. The **Attachment Editor** dialog box displays.
2. In **Name**, type a descriptive file name or title to identify the attachment.
3. In the text box, enter descriptive content for the body of the attachment and apply the formatting of your choice. You can also insert an image in the attachment and link to a URL. You can optionally remove the formatting and create the note in plain text format.
4. Click **OK**.

To edit an attachment

1. Select an HTML file in the **Attachments** list, and click **Edit HTML**.
2. In the **Attachment Editor** dialog box, apply your edits and click **OK**.

To copy an attachment

1. Select a file in the **Attachments** list, and click **Save As**.
2. In the **Save As** dialog box, type a new name for the file.
3. Click **Save**.

4. Click **OK**.

To delete an attachment

1. Select the file you want to remove on the **Attachments** page and click **Remove**.

   The attachment disappears from the list.
2. Click **OK**.

Customizing Tasks

If your administrator has created custom fields for the task component, you may be required to complete these fields when entering or modifying tasks or subtasks. The availability of custom fields depends upon your application license.

To set the values for custom properties:

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:

   - Double-click a task or sub-task in the **Task** pane.
   - Select a task or sub-task in the **Task** pane and choose **Task** > **Properties** .
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. Double-click a custom property on the **Custom** tab to open the **Edit Property** dialog box.
4. Type or select a new value for the property by double-clicking on the field:

integer, text, and real fields	**Value** is a text box.
enumerated types and user IDs	**Value** is a list box.
dates and times	**Value** has a **Date** check box and a **Time** check box, each of which is followed by a date or time in the format for your locale.

Tip: To enter a blank value for a **GroupList** or **UserList** property, click on a selected row to deselect it. When the item is no longer highlighted, click **OK**.

5. Click **OK**.

Adding Notes to Tasks

When you use the task component, you can enter additional information about the task or sub-task on the **Notes** tab in the **Task Properties** dialog box.

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:

   - Double-click a task or sub-task in the **Task** pane.
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. Click the **Notes** tab and type notes about the task in the **Notes** field.
4. Click **OK**.

Assigning Task Resources

As you create a task or sub-task, you can assign additional team members as resources to assist in the completion of the task. Use the **Resources** tab to review the list of team members available for this task assignment.

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:
   - Double-click a task or sub-task in the **Task** pane.
   - Select a task or sub-task in the **Task** pane and choose **Task** > **Properties** .
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. Select the **Resources** tab and click **Add**.

   The **Select Task Resources** dialog box opens.
4. Select the team members you want to assign to the task in the **Users** list, and click **Add**.

   The selected names are moved from the **Users** list to the **Assigned Resources** list.
5. Click **OK**.

Removing Task Resources

As you create a task or subtask, you may need to remove team members as resources to assist in the clean up of the task resources. Use the **Resources** tab to review the list of team members available for this task assignment.

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:
   - Double-click a task or sub-task in the **Task** pane.
   - Select a task or sub-task in the **Task** pane and choose **Task** > **Properties** .
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. Click the **Resources** tab and select a resource name from the **Task Resource Assignments** list.
4. Click **Remove**.
5. Click **OK**.

Note: You can also remove a team member from the **Assigned Resources** list in the **Select Task Resources** dialog box by selecting a name in the **Assigned Resources** list and clicking **Remove** to return it to the **Users** list.

Estimating Tasks

The task component includes a **Time** tab on which you can record the amount of time needed to complete a task or sub-task.

To enter the estimated time to complete a task

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:
   - Double-click a task or sub-task in the **Task** pane.

- Select a task or sub-task in the **Task** pane and choose **Task** > **Properties** .
- Right-click a task or sub-task in the **Task** pane and choose **Properties**.

The **Task Properties** dialog box displays.

3. Click the **Time** tab.
4. Use the **Start** and **Finish** buttons to select a start and finish date.
5. Type the estimated hours required to complete the task in the **Work** field.

   The rest of the **Time** pane is disabled, however, the values for **Actual** and **Variance** automatically calculate when the **Work** value changes.
6. Click **OK**.

Adding Comments to Task Revisions

After you modify one or more properties in the **Task Properties** dialog box and click **OK**, the application creates a new revision of the task. You should add a revision comment or note explaining why you made the revision prior to clicking **OK** in the dialog box.

1. Double-click a task in the upper pane to open the **Task Properties** dialog box.
2. Make the desired changes to the task properties.
3. Click the **Comment** tab and type a revision comment.
4. Click **OK**.

Marking Item Threads Read or Unread

You can display requirement, task, and topic components in either a list display or hierarchical format. StarTeam provides a default view for requirement, task, and topic components in a hierarchical structure. When you select to view the hierarchical structure, each new item becomes the root of a tree. Its branches are child requirements, subtasks to the parent task, or responses to the topic. Children of children requirements, subtasks of subtasks, and responses to responses form additional branches. When an item thread is unread, the textual information about the item thread displays in bold text in upper pane. When an item thread is read, the item thread displays in normal text in the upper pane.

The main menu or context menu commands enable you to mark item threads as read or unread to better track these components.

1. Select a topic thread in the upper pane.

   Tip: To select multiple threads, display the information in list format by clicking the **List Display** button in the toolbar.
2. Do one of the following:
   - Choose **Mark Thread as Read** to remove the bold format for the item thread.
   - Choose **Mark Thread as Unread** to add bold format for the item thread.

Updating a Group of Items

The server configuration settings must allow bulk update in the workflow rules.

There are times when you may want to update several items at the same time. Items that you can edit in a group are Requirements, Change Requests, Tasks, and etc.

1. Navigate to and select the items you want to edit.
2. Right-click and choose **Bulk Update**. The dialog appropriate to the items appears.
3. Make updates to fields.

4. Click **OK**. The selected items are updated.

Working with Work Records in Tasks

After working on a task or subtask, you should add a work record to indicate what was done and the time spent. For example, if you work on a task for one hour on Day one and for three hours on Day two, you would enter two work records, one for each day. You can edit or delete previously entered work records

Important: After you have added a work record to a task, you cannot create subtasks for that task.

Adding a Work Record to a Task

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:

   - Double-click a task or sub-task in the **Task** pane. The **Task Properties** dialog box displays. Click **Add** on the **Work** page.
   - Select a task and choose **Task** > **Add Work** .

   The **Work Record** dialog box opens and displays your **User Name** in the list at the top.
3. Click the **Date** button and select a date for the work record.
4. Type the number of hours worked in the **Work** field.
5. Type the number of hours it will take to complete the task in the **Remaining Work** field.
6. Type comments about the progress that has been made in the **Comments** field.
7. Click **OK**.

Editing a Work Record for a Task

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:

   - Double-click a task or sub-task in the **Task** pane.
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. On the **Work** page, select a record from the **Work Records** list and click **Edit**. This opens the **Work Records** dialog box.
4. In the **Work Records** dialog box, make any changes to the work record.
5. Click **OK**.

Deleting a Work Record from a Task

1. Select a folder from the folder tree and click the **Task** tab in the upper pane.
2. Do one of the following:

   - Double-click a task or sub-task in the **Task** pane.
   - Right-click a task or sub-task in the **Task** pane and choose **Properties**.

   The **Task Properties** dialog box displays.
3. On the **Work** page, select a record from the **Work Records** list and click **Delete**.
4. The message, `Delete Work Record?` appears. Click **Yes** to confirm the deletion.
5. Click **OK**.

Task Properties

This topic presents the task properties and their descriptions as displayed in the **Task Properties** dialog box. The **Task Properties** dialog box contains the following tabbed pages of properties.

Task

The following properties are on the **Task** tab.

Subtask Of Displays the name of the task for which this item is a subtask (if this item is a subtask).

Name Displays the name of the task or subtask.

Responsibility Displays the name of the person responsible for the completion of this task or subtask. Other people can be assigned as additional resources.

Milestone Indicates that the task or subtask should be treated as a milestone.

Status Displays the task status:

- **Pending** Waiting for completion of a predecessor task.
- **Ready To Start** Work can be started on the task.
- **In Progress** Work has been entered for the task.
- **Finished** Work is finished on the task.
- **Closed** Task is completed and closed.
- **Hold** Work temporarily stopped on the task, usually to wait for completion of another task.

Priority Displays the task priority level. The default is **Medium**. These priorities are identical to those in MS Project. **Do Not Level** is a Microsoft Project-specific term you should ignore.

Duration Indicates the number of hours expected for completion of the task.

Percent Complete Displays the percentage of work that has been completed on a task.

Needs Attention Notifies team leaders or task reviewers that this task requires attention. Enter the information about why this task needs attention in the text box below the **Needs Attention** check box.

Resources

The **Resources** tab lists the task resources. You can assign responsibility to team members by adding them to the list with the **Add** button, and you can remove them using the **Remove** button.

Time

The following properties are on the **Time** tab.

Plan Start Displays the start date for the task.

Plan Finish Displays the finish date for the task.

Plan Work Indicates the number of hours estimated to complete this task.

Actual Start Displays the actual start date calculated from work record entry.

Actual Finish Displays the actual finish date task status changes to **Finished**.

Actual Work Indicates the actual number of hours taken to complete the task, calculated from Work Records.

Variance Start Displays the variance in days between expected start date and actual start date. This is read-only calculated value.

Variance Finish Displays the variance in days between expected finish date and actual finish date. This is read-only calculated value.

Variance Work Displays the variance in number of hours between estimated and actual duration. This is read-only calculated value.

Work

The **Work** tab lists all the work records entered for this task. Each work record has the following properties.

User Name Displays the name of person who performed the work for this work record entry.

Date Displays the date of work record entry.

Work Hours Indicates the number of hours worked for this work record entry.

Remaining Work Indicates the remaining number of hours left to complete the task.

Comments Displays text comments explaining what work was done for this work record.

Total Actual Work Displays read-only calculated field of the total time spent on this task based on the work records entered.

Notes

The **Notes** tab is a simply a text box for capturing notes about the task.

Custom

You can create custom properties for an item which will display in the item **Properties** dialog box.

The following properties are on the **Custom** tab.

Property Displays each custom property name.

Value Displays the values for each custom property. Double-click the property name to edit the value.

Attachments

The **Attachments** tab contains a list of all the files attached to the current item.

Comment

The following properties are on the **Comment** tab.

Comment For This Revision Displays the reason for the changes to the current revision.

Comment For New Revision Displays the reason for the changes to the new revision.

Task Fields

This section lists all the task fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or the StarTeam Notification Agent. Reports can use any field name.

Actual Finish

Values: `date/time`.

Internal Identifier: `StTaskActualFinish`.

The actual finish date for a task.

Actual Hours

Values: `number`.

Internal Identifier: `StTaskActualHours`.

The number of hours spent completing the task.

Actual Start

Values: `date/time`.

Internal Identifier: `StTaskActualStart`.

The actual start date for a task.

Attachment Count

Values: `number`.

Internal Identifier: `AttachmentCount`.

The number of files attached to an item.

Attachment IDs (Advanced)

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. For example: `[00 00 00 00 02 00 00 00]` indicates two specific attachments.

Internal Identifier: `AttachmentCount`.

Cannot be used in queries. The ID numbers assigned to attachments. For example, the first attachment within a project is `00 00 00 00`.

Attachment Names

Values: `text` containing a series of file names separated by spaces.

Internal Identifier: `AttachmentNames`.

The names of the files attached to an item.

Attention Notes

Values: `text`.

Internal Identifier: `StTaskAttentionNotes`.

The text in the **Needs Attention** note.

Children Count

Values: `number`.

Internal Identifier: `ChildrenCount`.

The number of items that are children of this item. This is a client-calculated field.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those 2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created or the time at which the label associated with the promotion state was created.

Constraint Date

Values: `date/time`.

Internal Identifier: `StTaskConstraintDate`.

A task’s constraint date from MS Project.

Constraint Type

Values: `As Late As Possible, As Soon As Possible, Finish No Earlier Than, Finish No Later Than, Must Finish On, Must Start On, Start No Earlier Than, Start No Later Than`.

Internal Identifier: `StTaskConstraintType`.

A task’s constraint type from MS Project.

Created By

Values: list of users, <`None`>.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Deleted By

Values: list of users, <`None`>.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

Estimated Finish

Values: `date/time`.

Internal Identifier: `StTaskEstimatedFinish`.

The estimated finish date for a task.

Estimated Finish Variance

Values: `date/time`.

Internal Identifier: `StTaskEstimatedFinishVariance`.

The difference between the estimated and the actual finish date for a task.

Estimated Hours

Values: `number`

Internal Identifier: `StTaskEstimatedHours`.

The number of hours spent completing the task.

Estimated Hours Variance

Values: `number`.

Internal Identifier: `StTaskEstimatedHoursVariance`

The difference between the estimated and the actual number of hours spent completing the task.

Estimated Start

Values: `date/time`.

Internal Identifier: `StTaskEstimatedStart`.

The estimated start date for a task.

Estimated Start Variance

Values: `date/time`.

Internal Identifier: `StTaskEstimatedStartVariance`.

The difference between the estimated and the actual start date for a task.

Flag

Values: `No`, `Yes`.

Internal Identifier: `Flag`.

Marks or bookmarks files in the upper pane on your workstation. This is a client-calculated field.

Flag User List (Advanced)

Values: text displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `FlagUserList`.

Can be used in queries. Identifies users who have set flags on a given item.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

Is My Task?

Values: `No`, `Yes`.

Internal Identifier: `IsMyTask?`.

Indicates whether the logged on user is responsible for a task. This is a client-calculated field.

Is Replicated

Values: `0`, `1`.

Internal Identifier: `Is Replicated` (contains spaces).

Indicates whether the task is from MS Project task.

Item Deleted By

Values: list of users, `None`.

Internal Identifier: `ItemDeletedUserID`.

The name of the user who deleted this item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Item Deleted Time

Values: `date/time`.

Internal Identifier: `ItemDeltedTime`.

The time at which the item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Last MS Project Update

Values: `date/time`.

Internal Identifier: `StTaskMSProjectLastUpdate`.

The date that a task was last updated from MS Project.

Last Work/ Dependency Update

Values: `date/time`.

Internal Identifier: `StWorkDependencyLastUpdate`.

The last time that a work record or a dependency (task successor or predecessor) was added, edited, or deleted. This field is for use with MS Project.

Locked By

Values: list of users, <`None`>.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

Milestone

Values: `No`, `Yes`.

Internal Identifier: `StTaskMilestone`

Indicates whether a task represents a milestone. In MS Project, the definition for a milestone is a task of zero time length. It serves as a heading for one or more tasks to which a time length has been assigned.

In the application, a task has a milestone check box. After work is assigned to a task, it is no longer a milestone.

Modified By

Values: list of users, <`None`>.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

MS Project File Name (Advanced)

Values: `text`.

Internal Identifier: `StTaskMSProjectFileName`.

The name of the MS project file from which a task was exported.

MS Task GUID (Advanced)

Values: `text`.

Internal Identifier: `StTaskGUID`.

The GUID for a task in MS Project.

MS Task Unique ID (Advanced)

Values: `number`.

Internal Identifier: `StTaskUniqueID`.

The unique ID for a task in MS Project.

MS WBS Code (Advanced)

Values: `text`.

Internal Identifier: `StTaskWBSCode`.

A task's WBS code from MS Project.

My Lock

Values: `Exclusively Locked By Me`, `Non-exclusively Locked By Me`, `Not Locked By Me`.

Internal Identifier: `MyLock`.

Indicates whether the current user has the item locked and, if so, whether that lock is exclusive or not. This is a client-calculated field.

Needs Attention

Values: `No`, `Yes`.

Internal Identifier: `StTaskNeedsAttention`.

Indicates that the check box for **Needs Attention** has been selected.

New Revision Comment (Advanced)

Values: `text`.

Internal Identifier: `NewRevisionComment`.

Internal use only. The client uses this value during the item update process. The field always appears empty if added to the upper pane. This is a client-calculated field.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Notes

Values: `text`.

Internal Identifier: `Notes`.

Text comments on the effort levels for this item.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Percent Complete

Values: `number`.

Internal Identifier: `StTaskPercentComplete`.

A percentage indicating how much of a task has been completed.

Priority

Values: `Do Not Level, High, Higher, Highest, Low, Lower, Lowest, Medium, Very High, Very Low`.

Internal Identifier: `StTaskPriority`.

Indicates the priority given to a task. These priorities are identical to those in MS Project.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item's configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Read Status

Values: `Read`, `Unread`.

Internal Identifier: `ReadStatus`.

Indicates whether an item is considered read or not read. This is a client-calculated field.

Read Status User List

Values: `text` displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `ReadStatusUserList`.

Can be used in queries. Identifies users for whom a given item's status is **Unread**.

Resource Count

Values: `number`.

Internal Identifier: `StTaskResourceCount`.

The number of users listed as resources for a task.

Resource IDs (Advanced)

Values: `byte array`, displayed as a bracketed series of numbers in hex format. For example, `[14 00 00 00]` indicates a specific user.

Internal Identifier: `StTaskResourceIDs`.

Cannot be used in queries. The ID numbers assigned to the users who are this task's resources.

Resource Names

Values: `text` containing a series of user names separated by spaces

Internal Identifier: `StTaskResourceNames`

The names of the users who are this task's resources.

Responsibility

Values: list of users, <`None`>.

Internal Identifier: `StTaskResponsibility`.

The name of the user who is currently responsible for the task.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item's properties or contents. Additional text is stored in the **Comment** field.

Status

Values: `Closed, Finish, Hold, In Progress, Pending, Ready To Start`.

Internal Identifier: `STTaskStatus`.

Indicates the status of the task.

Task Duration

Values: number

Internal Identifier: `STTaskDuration`

The number of hours during which any user is working on a task. For example if two people will work eight hours on a task, the duration is eight hours if they work at the same time or a maximum of 16 hours if they do the work on different days.

Task Name

Values: `text`.

Internal Identifier: `STTaskName`.

The name of the task.

Task Number

Values: `number`.

Internal Identifier: `StTaskNumber`.

The number assigned to a task. For example, if the **Object ID** is `0`, the task number is `1`.

Task Origin

Values: `MSProject`, `StarTeam`.

Internal Identifier: `STTaskOrigin`.

Indicates whether the task was created in the application or exported to the application from Microsoft Project.

Task Type

Values: `Fixed Duration`, `Fixed Units`, `Fixed Work`.

Internal Identifier: `StTaskType`.

A task's type in MS Project.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is `2`.

Work Record Count

Values: `number`.

Internal Identifier: `WorkRecCount`.

The number of work records currently added to a task.

Topics

Topics are threaded conversations, that is, a series of messages that indicate how the messages are related. Each series of messages forms a tree with the initial message at its root. The topic component provides threaded conversations that you can place in specific project folders and link to specific project items. For example, you can link a topic to the change requests and file revisions that result from the topic discussion.

The upper pane of the client consists of topics and a series of responses to each topic. A series of topic trees are eventually formed, each of which consists of a root topic and its responses. The topic tree resembles a conversation that may go on among several people. In the client, this is called a threaded conversation because a topic and its responses are threaded together, starting with the root topic. By reading each response in a thread, one after the other, and the responses to those responses, you can see how the discussion has evolved. A number of other operations can be performed on topics or responses such as moving or sharing them.

Historical Value of Topics

Topics can raise general questions about the project or start very specific discussions about issues, such as feature implementation. While the responses can lead to resolution of these issues, the historical value of these conversations to the project can be even more significant. Future team members can:

- Reassess decisions more capably.
- Avoid retrying solutions that were previously found faulty.
- Understand why a particular solution to a problem became necessary and, therefore, not replace that solution with one that does not meet all the necessary criteria.

How Topics Can Help

Any type of threaded messaging improves teamwork on product development. However, because StarTeam has tightly integrated components, it enables team members to:

- Search topics and responses for specific words or phrases.
- Sort topics and responses.
- Filter topics and responses.
- View relationships between topics and their responses.
- Move and share topics (from the tree format).
- Link topics directly to folders or other items, such as change requests.
- Ask questions and quickly receive input while working on a file.
- Attach notes to a topic explaining why a particular method was used.
- Point out aspects of the project that may need to change in a later release.

Creating Topics

To start a threaded conversation, you must first create a topic.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Select the **Topic** tab.
3. Click **New** > **Topic** . The **New Topic** dialog box opens.
4. Click the **Topic** tab and type the title of your topic in the **Title** field.
5. Type the content for this topic in the **Content** field.
6. Use the **Options** tab if you want to send the topic to specific team members, assign a priority to the topic, or indicate a status for the topic.
    1. Select the **Options** tab.
    2. Click **Add**. The **Select Topic Recipients** dialog box opens.
    3. Select the team members from the list, then click **Add**.
    4. To assign a priority to the topic, select **Low**, **Normal**, or **High** from the **Priority** list.
    5. To specify a topic status, select either **Active** or **Inactive** from the **Status** list. The default status is **Active**.
7. If your administrator created additional topic properties, you can access them on the **Custom** tab.
    1. Double-click a custom property on the **Custom** tab to open the **Edit Property** dialog box.
    2. Type or select a new value for the property by double-clicking on the field:

integer, text, and real fields	**Value** is a text box.
enumerated types and user IDs	**Value** is a list box.
dates and times	**Value** has a **Date** check box and a **Time** check box, each of which is followed by a date or time in the format for your locale.

Tip: To enter a blank value for a **GroupList** or **UserList** property, click on a selected row to deselect it. When the item is no longer highlighted, click **OK**.

    3. Click **Apply**.
8. Use the **Attachments** tab if you want to attach a file, note, or graphic with your topic.
9. Optionally, select the **Comment** tab to add additional notes or a comment in the **Comment for new revision** field.
10. Click **OK**. This action enters the new topic in the upper pane of the **Topics** component.

If the tree format is selected, the topic title, your user name, and the time stamp display. If the list format is selected, the list displays the same information, but includes one additional column, **Description**, which shows the first few words in the topic text.

Working with Attachments

You can attach image files, HTML/plain text formatted notes, and other file types to a component using the **Attachments** tab on the **Properties** dialog box of the selected component. The HTML/plain text attachment is similar to a note in which you can easily format and align text without having to use an external editor.

Note: Attachments can only be used with Change Request, Requirement, Task, and Topic components.

1. Select a folder from the folder hierarchy and click the component tab in the upper pane (for example, **Change Request**) .

2. Open a component item that you want to add an attachment to, delete an attachment, or edit an existing attachment.
3. Click the **Attachments** tab.

To add an attachment

1. Click **Add**. The **Open** dialog box displays.
2. Select the file you want to attach to the component and click **Open**. To select multiple items, use the keyboard functions **Shift+Click** or **Ctrl+Click.** The selected item appears in the **Attachments** list on the **Attachments** page of the **Properties** dialog box.
3. Click **OK**.

To add an HTML or plain text attachment

1. On the **Attachments** tab, click **Add HTML**. The **Attachment Editor** dialog box displays.
2. In **Name**, type a descriptive file name or title to identify the attachment.
3. In the text box, enter descriptive content for the body of the attachment and apply the formatting of your choice. You can also insert an image in the attachment and link to a URL. You can optionally remove the formatting and create the note in plain text format.
4. Click **OK**.

To edit an attachment

1. Select an HTML file in the **Attachments** list, and click **Edit HTML**.
2. In the **Attachment Editor** dialog box, apply your edits and click **OK**.

To copy an attachment

1. Select a file in the **Attachments** list, and click **Save As**.
2. In the **Save As** dialog box, type a new name for the file.
3. Click **Save**.
4. Click **OK**.

To delete an attachment

1. Select the file you want to remove on the **Attachments** page and click **Remove**.

   The attachment disappears from the list.
2. Click **OK**.

Responding to Topics or Responses

After someone starts a topic, you can reply to the topic or to one or more of its responses.

1. On the **Topic** tab in the upper pane, select the item to which you are responding.
2. Right click the topic and select **Respond**. The **New Topic** dialog box opens.
3. On the **Topic** tab in the **New Topic** dialog box, type the title of your response in the **Title** field.
4. Type your remarks in the **Content** field.
5. Use the **Options** tab if you want to send the topic to specific team members, assign a priority to the topic, or indicate a status for the topic.

   1. Select the **Options** tab.
   2. Click **Add**. The **Select Topic Recipients** dialog box opens.
   3. Select the team members from the list, then click **Add**.
   4. To assign a priority to the topic, select **Low**, **Normal**, or **High** from the **Priority** list.
   5. To specify a topic status, select either **Active** or **Inactive** from the **Status** list. The default status is **Active**.
6. If your administrator created additional topic properties, you can access them on the **Custom** tab.

    1. Double-click a custom property on the **Custom** tab to open the **Edit Property** dialog box.
    2. Type or select a new value for the property by double-clicking on the field:

integer, text, and real fields	**Value** is a text box.
enumerated types and user IDs	**Value** is a list box.
dates and times	**Value** has a **Date** check box and a **Time** check box, each of which is followed by a date or time in the format for your locale.

       Tip: To enter a blank value for a **GroupList** or **UserList** property, click on a selected row to deselect it. When the item is no longer highlighted, click **OK**.
    3. Click **Apply**.
7. Use the **Attachments** tab if you want to attach a file, note, or graphic with your topic.
8. Optionally, select the **Comment** tab to add additional notes or a comment in the **Comment for new revision** field.
9. Click **OK**. This action enters the new response as a child of the topic.

If the tree format is selected, the response appears in relation to other parts of the threaded conversation. The response title, your user name, and the time stamp also display. If the list format is selected, the list displays the same information, but includes one additional column, **Description**, which shows the first few words in the response text.

Marking Item Threads Read or Unread

You can display requirement, task, and topic components in either a list display or hierarchical format. StarTeam provides a default view for requirement, task, and topic components in a hierarchical structure. When you select to view the hierarchical structure, each new item becomes the root of a tree. Its branches are child requirements, subtasks to the parent task, or responses to the topic. Children of children requirements, subtasks of subtasks, and responses to responses form additional branches. When an item thread is unread, the textual information about the item thread displays in bold text in upper pane. When an item thread is read, the item thread displays in normal text in the upper pane.

The main menu or context menu commands enable you to mark item threads as read or unread to better track these components.

1. Select a topic thread in the upper pane.

   Tip: To select multiple threads, display the information in list format by clicking the **List Display** button in the toolbar.
2. Do one of the following:
   - Choose **Mark Thread as Read** to remove the bold format for the item thread.
   - Choose **Mark Thread as Unread** to add bold format for the item thread.

Topic Properties

This topic presents the topic properties and their descriptions as displayed in the **Topic Properties** dialog box. The **Topic Properties** dialog box contains the following tabs of properties.

Topic

The following properties are on the **Topic** tab.

Title	Displays the title of the topic.
Created By	Displays the name of person who created the topic.

Created On Displays the date on which topic was created.

Attachments Displays the number of attachments to the topic.

Modified By Displays the name of the last person who modified the topic.

Modified On Displays the date on which the topic was last modified.

Content Displays the text contents of the topic.

Options

The following properties are on the **Options** tab.

Recipients Displays the list of intended recipients of the topic or response.

Note: You cannot delete yourself as a recipient unless you delete all the recipients. When there are recipients, StarTeam does not allow you to remove yourself from the notification list.

Priority Displays the topic priority: **Low**, **Normal**, or **High**

Status Displays the topic status: **Active** or **Inactive**.

Custom

You can create custom properties for an item which will display in the item **Properties** dialog box.

The following properties are on the **Custom** tab.

Property Displays each custom property name.

Value Displays the values for each custom property. Double-click the property name to edit the value.

Attachments

The **Attachments** page contains a list of all the files attached to the current topic.

Comment

The following properties are on the **Comment** tab.

Comment For This Revision Displays the reason for the changes to the current revision.

Comment For New Revision Displays the reason for the changes to the new revision.

Topic Fields

This section lists all the fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or Notification Agent. Reports can use any field name.

Am I Recipient?

Values: `No`, `Yes`.

Internal Identifier: `AmIRecipient?`.

Indicates whether the logged on user is a recipient of a topic. This is a client-calculated field.

Attachment Count

Values: `number`.

Internal Identifier: `AttachmentCount`.

The number of files attached to an item.

Attachment IDs (Advanced)

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. For example: `[00 00 00 00 02 00 00 00]` indicates two specific attachments.

Internal Identifier: `AttachmentCount`.

Cannot be used in queries. The ID numbers assigned to attachments. For example, the first attachment within a project is `00 00 00 00`.

Attachment Names

Values: `text` containing a series of file names separated by spaces.

Internal Identifier: `AttachmentNames`.

The names of the files attached to an item.

Children Count

Values: `number`.

Internal Identifier: `ChildrenCount`.

The number of items that are children of this item. This is a client-calculated field.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those 2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created or the time at which the label associated with the promotion state was created.

Content

Values: `text`.

Internal Identifier: `Description`.

The text of a topic.

Created By

Values: list of users, `<None>`.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Deleted By

Values: list of users, `<None>`.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

Flag

Values: `No`, `Yes`.

Internal Identifier: `Flag`.

Marks or bookmarks files in the upper pane on your workstation. This is a client-calculated field.

Flag User List (Advanced)

Values: text displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `FlagUserList`.

Can be used in queries. Identifies users who have set flags on a given item.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

Item Deleted By

Values: list of users, `None`.

Internal Identifier: `ItemDeletedUserID`.

The name of the user who deleted this item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Item Deleted Time

Values: `date/time`.

Internal Identifier: `ItemDeltedTime`.

The time at which the item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Locked By

Values: list of users, `<None>`.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

Modified By

Values: list of users, `<None>`.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

My Lock

Values: `Exclusively Locked By Me`, `Non-exclusively Locked By Me`, `Not Locked By Me`.

Internal Identifier: `MyLock`.

Indicates whether the current user has the item locked and, if so, whether that lock is exclusive or not. This is a client-calculated field.

New Revision Comment (Advanced)

Values: `text`.

Internal Identifier: `NewRevisionComment`.

Internal use only. The client uses this value during the item update process. The field always appears empty if added to the upper pane. This is a client-calculated field.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Priority

Values: `High`, `Low`, `Normal`.

Internal Identifier: `Priority`.

The value of the **Priority** field. You can use repository customization to change the names of these values or include other values.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item's configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Read Status

Values: `Read`, `Unread`.

Internal Identifier: `ReadStatus`.

Indicates whether an item is considered read or not read. This is a client-calculated field.

Read Status User List

Values: `text` displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `ReadStatusUserList`.

Can be used in queries. Identifies users for whom a given item's status is **Unread**.

Recipient Count

Values: `number`.

Internal Identifier: `RecipientCount`.

The number of recipients to whom a topic is addressed.

Recipient IDs

Values: `byte array`, displayed as a bracketed series of numbers in hex format. For example, `[14 00 00 00]` indicates a specific user.

Internal Identifier: `RecipientIDs`.

Can not be used in queries. The ID numbers assigned to the users who are recipients (people to be notified about this topic).

Recipient Names

Values: `text` containing a series of users names separated by spaces.

Internal Identifier: `Recipient Names`.

The names of the recipients designated for notification about this topic.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item's properties or contents. Additional text is stored in the **Comment** field.

Status

Values: `Active`, `Inactive`.

Internal Identifier: `Status`.

Indicates the status of this topic.

Title

Values: `text`.

Internal Identifier: `Title`.

The text of the **Title** field.

Topic Number

Values: `number`.

Internal Identifier: `TopicNumber`.

The number assigned to a topic. For example, if the **Object ID** is `0`, the topic number is `1`.

Type

Values: `Response`, `Topic`.

Internal Identifier: `Type`.

Indicates whether the item is a topic (root of a topic tree) or a response (branch of a topic tree). This is a client-calculated field.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is `2`.

Files

To place a file under version control, it must be added to a folder in a StarTeam project view, which stores a copy of the file in the StarTeam repository. After the file has been added to StarTeam, you and other members of your team can check it out, revise it, and check in new revisions, while StarTeam maintains information on all revisions of the file. Note that all check-ins in StarTeam are atomic.

When checking out a file revision, you should verify that you have the *tip* or latest version of the file. Doing this ensures that the file you see contains the latest changes. If you intend to modify the file, you should check it out with an exclusive lock, to indicate to others that you are working on it.

When you check a file in, StarTeam records the file changes as a new revision. As part of the check-in process, you can remove the lock, notifying others that the file is available, or maintain the lock, showing that you intend to continue working on the file. If two team members change the same text file simultaneously or if one member changes an outdated file, StarTeam contains a merge option that allows the file changes to be combined so that no work is lost. In such cases, StarTeam assigns a *Merge* status to the file.

Note: The SDK, StarTeam Server, and most clients support files larger than 4 GB. If you plan on taking advantage of large file support, you should upgrade all users to the current StarTeam version. Large file sizes are not compatible with older StarTeam versions.

Files Under Version Control

If a file resides in the working folder of an application folder, you can add that file to the application folder. This operation places that file under version control. A copy of the working file becomes the first revision of that file stored in the repository. If the working file is deleted later, the data is not lost because a copy exists in the repository. The application creates a new revision of this file in the repository every time you check the file in.

Every time you check a file revision out, its contents are copied to a working folder. Checking out a revision also ensures that you have the tip or a specific revision to work on. For example, you may need a team member’s most recent changes to a file, or you may have deleted the working file from your hard drive and now need another copy.

The application enables you to label the tip revisions of every item within a view. For example, when the project reaches a particular milestone (such as beta), you might give the view’s items a label, called a view label. Then you can configure the view to return to the way it was at the time the label was applied, check out revisions as a group using that label, create a new view based on the label, or assign the label to a promotion state.

The application also provides revision or version labels. You can label one or more revisions as you check them in or by applying the label to each of the revisions using the **Labels** command on the File menu. StarTeam makes it easy to check out those files as a group using the label. A file revision can have any number of labels. However, no two revisions of the same file in the same view can have the same label.

Recommendations for Working with Files Under Version Control

Here are some recommendations about using files under version control:

- To let other team members know that you intend to make changes to a file, change the lock status to *exclusive* as part of the check-out procedure.
- As part of the check-in process, you can notify others both that you are finished making your changes to the file and that it is available for them to check out by removing the lock status.
- If you intend to continue making changes to the file but still want to check it in for backup purposes, keep the file locked.
- If two team members change the same text file simultaneously or if one member changes an outdated file, you can use the merge option to combine the changes in these files so no work is lost. In such cases, the application gives the file a `Merge` status.
- To prevent yourself from changing a file that you have not locked, select the **Mark Unlocked Working Files Read-only** personal option. Then, if you check out a file that you have not locked, the working copy becomes read-only.

Adding Files to Projects

You can add files to a project folder if the files are in the correct working folder.

1. Put the files you want to add to the project into the local working folder that corresponds to the repository folder in StarTeam.
2. Open the project view in StarTeam and select the folder to which you want to add the files.
3. Select **Files Not In View** in the **Filter** list on the toolbar. This displays only the files in your working folder that have not been added to the project view.
4. Select the file(s) you want to add to the project, and choose **Files** > **Add Files** .
5. Type a generic description for all the files in the **Add Files** dialog box, or check **Prompt for description for each file**.
6. Select an appropriate lock status for the files. The default status is **Unlock**.
7. Check **Link and pin process item** to link the new files to process items, if process rules are enforced.
8. Check the **Mark selected process item as fixed/finished/complete** check box, if work on the active process item is final.
9. Check the **Show property editor for selected process item** check box to make changes to the selected process item's properties during the add process.
10. Optionally select a revision label from the list for the **Revision Label** field, or create a new revision label by typing the label name.

    Existing labels list in reverse chronological order, based on the time at which they were created.

11. Click **OK**.

Note: You can also click **Advanced** to select advanced options, such as performing an EOL conversion and selecting a file encoding. EOL conversion is based on the **EOL** settings in the **Personal Options** dialog box.

When you add new files to the project, the status for these files changes from **Not In View** to **Current**.

If you check **Delete working files** during check in, the status of the new files changes to **Missing**.

Opening and Editing Files

You can open a file directly from within StarTeam if the file has a status of **Current**. You can also display a file in the default editor, Notepad, or an alternate editor. If the file is an executable, such as a `.bat` file, or has no associated application, you must use the **Edit** command to edit the file from the application.

When you open a file in StarTeam, the application does one of the following:

- Runs the file if it is an executable, such as `autoexec.bat`.
- Displays an error messages if the file is not an executable and there is no associated application.
- Opens the file in an associated application. For example, `.doc` files will open in Microsoft Word. However, this only works if an open action file association exists for the selected file’s extension in the operating system.

Opening a File

1. Click the **File** tab.
2. Choose **File** > **Open** .

If the file does not open in an associated application, an association may not have been created for the selected file type. See your operating system documentation for instructions on associating file types with applications.

Editing a File

1. Click the **File** tab.
2. Choose **File** > **Edit** .

Note: The file opens in the default editor, Notepad, unless you set an alternate editor.

Changing the Default Editor

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Click the **File** tab.
3. Click **Alternate Applications**. The **Alternate Applications** dialog box opens.
4. Check **Editor** and browse to the executable file for the editor you want as the default application.
5. Type any appropriate **Options**.
6. Click **OK**.

Note: For non-Microsoft Windows systems, specify a command to use for launching files in an alternate application in the **Open with...** field on the **File** or **Folder** tab in the **Personal Options** dialog box.

Enabling Concurrent File Editing

The personal option named **Use Non-Exclusive Locks In Integrations** affects how files are locked when accessed from application integrations such as Visual Studio. If you select this option, locking a file (for example, as part of a check-out operation) creates a non-exclusive lock rather than an exclusive lock.

With an exclusive lock, only the person who has the file locked can check in the file. With a non-exclusive lock, others can check in the file. Exclusive locks are the safest, but non-exclusive locks are often preferred because text files can be easily merged using File Compare/Merge. Using non-exclusive locks allows more than one person to edit a file at one time. If team members are not editing the same lines of the file, the merged file usually has no conflicts.

If you are using an application integration for your development environment, for example, the integration with Visual Studio, you cannot check in files from the development environment if both the **Require Exclusive Lock When Files Are Checked In** check box in the **Project Properties** dialog box, (**Options** tab) and the **Use Non-exclusive Locks In Integrations** check box in the **Personal Options** dialog box (**Files** tab) are checked. The administrator usually determines the setting of the **Require Exclusive Lock When Files Are Checked In** check box. However, personal options are set by you for your workstation.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Click **Files**.
3. Check **Use Non-exclusive Locks In Integrations**.
4. Click **OK**.

Note: If you have checked **Use Non-exclusive Locks In Integrations** and experience check-in problems, try clearing the check box. You may want to talk to your administrator about the setting for the **Require Exclusive Lock When Files Are Checked In** check box.

Excluding Files from a Project

Some types of files will never be added to a project, although they may reside in a working folder. For example, suppose you are creating files with an application that makes an automatic backup (`.bak`) copy of each file every time you save the file. Although your working folder might contain several `.bak` files, you would have no reason to check them into (or out of) the application. Therefore, you should exclude them from the project view.

Exclude lists can also be inherited from parent folders.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.

   Note: You can also exclude files in **Not-in-View** folders, but you must have the root folder selected to do so.
2. Right-click the selected folder and choose **Properties** to open the **Folder Properties** dialog box.
3. Select the **Exclude** tab.

   Note: The **Exclude** tab does not affect files that are already part of the project.
4. Do one of the following:

Inherit and Use Local Exclude List	Excludes files that match the exclude list specifications set for this folder as well as those of its parent folder. If the **Local Exclude List** text box does not yet include any file specifications, add them.
Use Local Exclude List	Excludes files that match the exclude list specifications set for this folder. If the **Local Exclude List** text box does not yet include any file specifications, add them.

No Exclude List	Includes all files.

5. Type one or more file specifications to use for matching files.

   Use standard expressions (with `*` and `?` wild cards) separated by commas, spaces, or semicolons. To include a comma, space or semicolon as part of the specification, enclose the specification in double quotes.

   A trailing `/` character means that **Not-in-View** folders will be excluded. For example, `bin/` would cause all **Not-in-View** folders named `bin` to be excluded from the folder tree.

   Note: The `\` character does not work. It is treated as an escape character.

Finding Files Associated with Active Process Items

When you have files associated with an active process item, you can quickly find all associated file changes by following these steps.

1. Open the pane that contains the active process item.

   You can see what item is the active process item by looking at the left side of the **Status Bar**. The second box in the **Status Bar** displays the **Active Process Item** icon, followed by the name of the item.
2. Select the active process item and click the **Change** tab in the lower pane to see all of the workspace (check-in) change packages for which the selected active process item was used.

Hiding Folders and Files

Using the **Folder Properties** dialog box, you can set the **Visible** property to exclude folders and their files from visibility.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Double-click the folder. The **Folder Properties** dialog box opens.
3. Select the **Name** tab and uncheck the **Visible** option. This hides the folder and the files it contains
4. Click **OK**.

Note: To make the folder visible again, check the **Visible** option in the **Folder Properties** dialog box.

Marking Unlocked Files Read-only

In many cases, users make edits before realizing that their files must be exclusively or non-exclusively locked to check them in. If the files are read-only, users are less likely to make this mistake.

1. Choose **Project** > **Properties** . .

   The **Project Properties** dialog box opens.
2. Select the **Options** tab.
3. Check **Mark Unlocked Working Files Read-only**, which applies to files that are unlocked in the application or in application integrations with third-party applications.

   If this check box is cleared, you must use the operating system to change the read-only attribute to read/write.

   Working copies of unlocked files will now become read-only when the following file operations are performed:

   - File check-ins.
   - File check-outs (from the **File** or **History** pane).

- File unlocks.

Note: This project property overrides the identical **Mark Unlocked Working Files Read-only** personal option. If you change your mind after selecting the property (or the equivalent personal option), verify that no files are writable before clearing the check box. Next, force a check out and lock all the files (or just the read-only files). Finally, unlock them.

Renaming Files

If you want to rename a file in your project, you should rename it within the StarTeam application. This retains the properties associated with that file, such as history and links.

If you rename a file outside the application (for example, by using Windows Explorer), the application considers the file to be a new file. When you add the file with the new name, it will have no connection to the history, links, or other properties of the original file.

1. Select a file.
2. Do one of the following:
   - Choose **File** > **Properties** .
   - Right-click the selected file and choose **Properties**.
   - Click the **Properties** toolbar button.
3. Click the **General** tab in the **File Properties** dialog box.
4. Change the **Name**.
5. Click **OK**.

StarTeam renames the file in both the StarTeam repository and in your working folder.

Selecting Linked Files

You can quickly select all the files associated with a linked item directly from the component tab in the upper pane. There are two choices for selecting the linked files:

- You can select only the files linked to an item.
- You can add the files linked an item to an existing selection of files on the **File** pane.

1. Click the component tab in the upper pane containing the item to which the files are linked, and select the item.
2. Right-click the selected item and choose **Linked Files** > **Add To Selection** .

StarTeam switches the upper pane to the **File** pane, activates the **All Descendants View**, and adds all the files linked to the item to the existing file selection.

Note: Selected items must have linked files in the current view to perform this operation.

Setting the File Executable Bit for UNIX

When you add a file from a UNIX operating system, the state of the executable bit is preserved by StarTeam. For each file, there is an **Executable** check box that becomes selected if the executable bit is set and becomes cleared if the bit is not set. Future check-out operations ensure that the executable bit for the checked-out file matches the setting of the **Executable** check box.

1. Select a file.
2. Click **File** > **Properties** . The **File Properties** dialog box opens.
3. On the **General** tab, check or uncheck **Executable**.

4. Click **OK**.

File Annotation Pane

The **File Annotation** pane shows historical information about changes made to any text file in StarTeam.

Using the File Annotation Pane

1. Choose **Tools** > **File Annotation** . The **File Annotation** pane opens. It will be enabled for every view window opened in the current session.
2. Select a text file from the **File** tab to view the tip revision's complete annotation information, or from the **History** tab to view changes made to a specific revision of the file.

   Note: Annotation information can also be viewed from the **Link** tab if the item is a text file.

Like the **File Compare** tool, the **File Annotation** tool displays the contents of the selected file in an embedded pane at the bottom of the main window. If you resize the pane, the client will automatically display the split pane bars in the same position in every open view window.

Line by line, annotation information shown in this initial view of the file includes:

- The name of the user who made each change.
- A link to each process item linked to the revision.
- The dot notation of the file revision in which each change was made, which is linked to a view of the actual changes that were made to the content in that revision.
- An ℹ **Info** icon that provides more details about the file revision and linked process items, such as the modified time or any comments.

Reviewing a Previous Version of a File

1. Click the dot notation link of the file revision you wish to review. A new page will open that shows, line by line, the changes that occurred in the selected revision of the file.
2. Use the icons described below to navigate the changes made to each version of the selected file you review.

⇦	Go to the previous page.
⇨	Go to the next page.
⤓	Go to the end of the file.
⤒	Go to the beginning of the file.
⇥	Go to the first changed line in the file.
⇤	Go to the last changed line in the file.
➡	Go to the next changed line in the file.
⬅	Go to the previous changed line in the file.
⊕	Denotes a line that has been added to the file. The text of added lines is green.
↻	Denotes each line that has been revised. The text of revised lines is blue, with additions highlighted in green and deletions highlighted in red with a strikethrough effect.
⊖	Denotes each line that has been deleted from the file. The text of deleted lines is red with a strikethrough effect.

Viewing Previous File Revisions

You can review the contents of a prior file revision in either the default editor or in the application for which the file type is registered.

1. Click the **File** tab.
2. Select a file.
3. In the **History** tab in the lower pane **History** view, select the specific revision you want to review.
4. Right-click the selected item to open the context menu and choose one of the following:

   View Revision Content Copies the revision to a temporary file and display it in the default editor (Notepad or the alternate editor specified in the **Personal Options**

   Open Revision Content Copies the revision to a temporary file and display it in the associated application.

   Note: The client creates the temporary files in the local temp directory on the system. For example, if working on a Microsoft Windows system, the temporary files are created in the `C:\Documents and Settings\<user>\Local Settings\Temp` directory. When you exit the client, the files are deleted from the system.

Setting File Storage Options

StarTeam stores status information for the files in your working folder in a central location on your workstation or in a child folder (named `.sbas`) of each working folder.

Setting File Status Storage for a Specific View

Normally, this view property defaults to the storage method that you have selected as a personal option. You can, however, select a storage method for a specific view. If you do so, this setting takes precedence over the personal option.

1. Choose **View** > **Properties** . The **View Properties** dialog box opens.
2. Select **Central** or **Per Folder** in the **File Status Repository** group.

   Central If several users all check files in and out of StarTeam from a shared working folder, file statuses are stored on each of their computers. Whenever a user makes a change to a file in the working folder, the status for that file is updated only on that user's computer. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files.

   Per Folder Most useful in the case where multiple users are sharing a working folder because it causes the statuses to be updated within the working folder itself. Everyone has access to those status changes and **Unknown** statuses do not occur.

Setting File Status Storage for all Files

You can set file status storage for all your files through a personal option setting that controls file status information for all your files. This setting applies to all files that are in views for which you have not set a file status property.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Click the **Files** tab.
3. Select **Central** or **Per Folder** option in the **File Status Repository** group.

Central If several users all check files in and out of StarTeam from a shared working folder, file statuses are stored on each of their computers. Whenever a user makes a change to a file in the working folder, the status for that file is updated only on that user's computer. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files.

Per Folder Most useful in the case where multiple users are sharing a working folder because it causes the statuses to be updated within the working folder itself. Everyone has access to those status changes and **Unknown** statuses do not occur.

Controlling How File Status Information is Stored

File status information about the files you are working on is stored on your workstation either in a central location or in a child folder (named `.sbas`) of each working folder.

You can set the file status property for a specific view. The view property defaults to the storage method that you selected as a personal option. When changed from that default, the view property take precedence over your personal option for the view.

Note: You can also set your **Personal Options** to control file status information for all your files, unless those files are in views for which you have set the view property for file status.

1. Choose **View** > **Properties** . The **View Properties** dialog box opens.
2. Click the **Name** tab.
3. Select the **Central** or **Per Folder** option button in the **Repository** group.

   The per-folder option is most useful in the special case where multiple users are sharing a working folder. For example, on a shared network drive.

   For example, suppose several users all check files in and out of a shared working folder. If these users have set the central repository option for file statuses, the statuses are stored on each of their computers. Whenever a user makes a change to a file in the working folder, the status for that file is undated only on that user’s computer. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files. Using the per-folder option causes the statuses to be updated within the working folder itself. Everyone has access to those status changes and **Unknown** statuses do not occur.
4. Click **OK**.

Tip: Select **Default (Central)** to return to using the **Personal Options** settings.

File Status Information

The operations you can perform on a particular file depend upon the status of the file. A successful check-in or check-out operation updates the status of your working file to Current, unless you check out a historical revision or delete the file. This list includes all status classes and shows their effects on check-ins and check-outs.

Current The tip revision of this file is in your working folder.

Check-ins: Does not affect the file revision in the repository.

Check-outs Does not affect the file revision on your workstation.

Deleted on Disk A file with this status functions as a filtered element of the Missing status, but includes additional information that the local file in your working folder was deleted.

Check-ins Not applicable. If a file has a Deleted on Disk status, it is not in your working folder, so it cannot be checked in.

Check-outs Copies the file to your working folder. If a file has a `Deleted on Disk` status, you will be asked if you want to check it out when you open it. You can also check it out manually.

You can only view this status in the **Items** pane and it is not a value option in a query if you choose **Status as a Field** condition. However, when you generate a report this status does display on the report.

Note: If you deleted the file in your working folder, you may also want to delete it on the server or check out the file in the project view to your working folder. Otherwise, files in your working folder and in the project view will be out-of sync.

Deleted on Server

A file with this status functions as a filtered element of the Not in View status, but includes additional information that the file on the server in the project view was deleted.

Check-ins Not applicable. Files with the `Deleted on Server` status cannot be checked in. However, you can add them to the project with the **Add Files** command.

Check-outs Not applicable. Files with the `Deleted on Server` status are not in the project view, so they cannot be checked out.

You can only view this status in the **Items** pane and it is not a value option in a query if you choose **Status as a Field** condition. However, when you generate a report this status does display on the report.

Note: If the file is deleted on the server, you should verify whether the file in your working folder is obsolete. To prevent any issues, determine the reason for the deleted file before you add the file back to the project view.

Ignore

Ignore acts as a note to self to ignore the marked files. It does not change any functionality within the files or within StarTeam. You can restore files back to their StarTeam status when you are done ignoring them. Right-click the file and select **Ignore** or **Restore**.

Check-ins Not applicable. Files with the `Ignore` status can be checked in. However, marking them as `Ignore` indicates that you do not want to work with the file.

Check-outs Not applicable. Files with the `Ignore` status can be checked out. However, marking them as `Ignore` indicates that you do not want to work with the file.

Missing

The file is not in your working folder.

Check-ins Not applicable. If a file has a `Missing` status, it is not in your working folder, so it cannot be checked in.

Check-outs Copies the file to your working folder. If a file has a `Missing` status, you will be asked if you want to check it out when you open it. You can also check it out manually.

Merge

The file in your working folder has been modified, but is not based on the tip (latest) revision of this file. This status usually occurs when you and another person have both been working on a file, but the other person has checked it in before you.

Check-ins Unless you force a check-in, File Compare/Merge starts. If the file is a text file, File Compare/Merge can reconcile the differences between it and the tip revision. If the file is binary, you may be able to merge the working version and the tip revision by using the application in which the file was created.

Check-outs Not allowed, unless you force the check-out. The file changes you have made in your working folder will be lost.

Modified Your working file has been altered and is based on the tip revision of this file.

Check-ins Unless someone else has the file locked, you can check in the file.

Check-outs Not allowed unless you force the check-out. Your changes will be lost if you check out the file, because it overwrites the file in your working folder.

Modified on Disk and Deleted on Server A file with this status functions as a filtered element of the Not in View status, but includes additional information that the local file in your working folder was modified, but the same file on the server in the project view was deleted.

Check-ins Not applicable. Files with the `Modified on Disk` and `Deleted on Server` status cannot be checked in. However, you can add them to the project with the **Add Files** command.

Check-outs Not applicable. Files with the `Modified on Disk` and `Deleted on Server` status are not in the project view, so they cannot be checked out.

You can only view this status in the **Items** pane and it is not a value option in a query if you choose **Status as a Field** condition. However, when you generate a report this status does display on the report.

Note: You changed a file in your working folder, but since that file was deleted in the project view you should determine if you need to also delete it or update the project view with your modified file.

Not in view The file is in your working folder, but not in the project view.

Check-ins Not applicable. Files with the `Not in View` status cannot be checked in. However, you can add them to the project with the **Add Files** command.

Check-outs Not applicable. Files with the `Not in View` status are not in the project view, so they cannot be checked out.

Out of Date The file in your working folder is an outdated revision of the file. If you need the current revision, you should check it out.

Check-ins Not allowed, unless you force the check-in. Checking in an `Out Of Date` file overwrites the changes made to the tip revision of the file with your older file revision.

Check-outs Checking out an `Out Of Date` file makes the file in your working folder `Current`.

Unknown The file in the working folder has the same name as a file in the view, but the file in the view has not been checked out from the repository. You may have copied it from another location.

Check-ins Not allowed, unless you force the operation. If the file status is `Unknown`, the consequences of this action are also unknown, except that the file in your working folder becomes the tip revision in the repository. Before forcing a

	check-in, try to identify the file by using **Update Status** with an MD5 checksum instead of a time stamp (see *File Options*). If you still cannot identify the file, compare the file in your working folder to the current tip revision.
Check-outs	Allowed, if you merge the file in your working folder with the tip revision. However, because the first revision acts as the ancestor file for the merge, many parts of the files may be in conflict. Instead of merging, you may prefer to force a check-out or check-in. If the file status is `Unknown`, the consequences of forcing a check-out are also unknown, except that this action also overwrites the file in your working directory. Before forcing a check-out, try to identify the file by using **Update Status** with an MD5 checksum. If you still cannot identify the file, compare the file in your working folder to the current tip revision.

File Access Rights

When you open the **File Access Rights** dialog box and select the **File** node, the rights apply only to the selected file.

File access rights are also available from the **Folder Access Rights**, **View Access Rights**, and **Project Access Rights** dialog boxes. In these cases, the rights cover all files in the selected folder, view, or project rather than an individual file. Only in these cases are the following container-level access rights available:

- Add files to a folder
- Share/move file out of a folder
- Change file behavior/configuration

The following sections describe the access rights that are available from the **File** nodes in the **Project Access Rights**, **View Access Rights**, **Folder Access Rights**, and **File Access Rights** dialog boxes.

Generic item rights

See item and its properties	See files in the files list (upper pane) and view file properties by selecting **File** > **Properties** .
Modify properties	Change the file properties. Modifiable properties include the archive/file name, description, executable bit setting (useful only for non- Windows platforms), compression, storage options, and custom properties. If used, an alternate property editor (APE) may restrict the properties that can be modified and the users who can modify them still further.
Delete from folder	Delete files from their folders.
Change item access rights	Change access rights for the files. If you change this setting, be sure that you remain one of the users who can change access rights.
See history	See file history in the history pane.
Perform maintenance	Change the revision comments for past revisions.
Set exclusive locks	Lock files exclusively.
Break exclusive locks	Remove someone else’s exclusive lock on the files.

Label rights

Attach/Adjust view labels Add a view label to the files. Move a view label from one revision to another. This right controls direct manipulation of view labels for the files at the item level. This right does not stop users from attaching a view label to the files when a view label is created.

Detach view label Remove a view label from the files. Be aware that if users can delete view labels, they can detach a view label from the files by deleting the view label from the view, regardless of the setting of this right.

Attach/Adjust revision labels Add a revision label to the files. Move a revision label from one revision to another. This right controls direct manipulation of revision labels for the files at the item level. It can stop users from checking in files with an attached revision label.

Detach revision labels Remove a revision label from the files. Be aware that if users can delete revision labels, they can detach a revision label from the files by deleting the revision label from the view, regardless of the setting of this right.

Link rights

See links See the links created for the files.

Create links Link the files to other items.

Modify links Change a link to the files.

Delete link Delete a link that affects the files.

File-specific rights

Check in file Check in the files.

Check out file Check out the files.

Generic item container rights

Add file to folder Add files to a folder, view, or project if its **File Access Rights** dialog box has this option. This right appears only on the **File Access Rights** dialog boxes associated with a folder, view, or project.

Share/Move out of folder Share files or move files that reside in a folder, view, or project if its **File Access Rights** dialog box has this option. This right appears only on the **File Access Rights** dialog boxes associated with a folder, view, or project. Be aware that the access rights set for any file that is moved or shared go with that file into the new folder.

Change behavior or configuration Change the branching ability and configuration of files that reside in a folder, view, or project if its **File Access Rights** dialog box has this option. This right appears only on the **File Access Rights** dialog boxes associated with a folder, view, or project.

File Properties

This topic contains the file properties and their descriptions as displayed in the **File Properties** dialog box. The **File Properties** dialog box contains the following tabbed pages of properties.

General

The following properties are on the **General** tab.

Name	Displays the name of the file.
Description	Displays the description of the file.
Status	Indicates the relationship between the copy of the file in your working folder and the tip revision in the repository.
Size	Displays the size of the tip revision of the file in bytes.
Last Modified By	Displays the name of the person who last modified the file.
Last Modified On	Displays the date on which the file was last modified.
File Time Stamp	Displays the time at which the file was last modified.
Locked Exclusively By	Displays he name of the user who has exclusively locked the file.
Locked Non-Exclusively By	The name of user who has non-exclusively locked a file.
EOL check-out format	Displays the options for EOL formatting for text files during check-out: **Client Defined**: Causes workstation default or per-checkout EOL conversion option to be used. **Fixed CR**, **Fixed LF**, and **Fixed CRLF**: Causes this EOL format to be used always; the workstation/check-out conversion option is ignored.
File type	Indicates which file type is selected: ASCII, Binary, or Unicode.
Executable	Indicates whether the executable bit should be set for a UNIX file.

Working File

The following properties are on the **Working File** tab.

Path	Displays the Path to the working folder for the file.
File Exists	Indicates whether the file exists in the working folder.
Size	Displays size of the file in the working folder in bytes.
Time Stamp	Displays the time that the working folder file was last modified.
Executable	Indicates whether the executable bit should be set for a UNIX file.

Custom

You can create custom properties for an item which will display in the item **Properties** dialog box.

The following properties are on the **Custom** tab.

Property	Displays each custom property name.
Value	Displays the values for each custom property. Double-click the property name to edit the value.

Comment

The following properties are on the **Comment** tab.

Comment For This Revision	Displays the reason for the changes to the current revision.
Comment For New Revision	Displays the reason for the changes to the new revision.

File Fields

This section lists all the file fields in alphabetical order.

Note: Client-calculated fields cannot be used in custom email notifications or StarTeam Notification Agent. Reports can use any field name.

Archive Format (Advanced)

Values: `Native-II`.

Internal Identifier: `ArchiveFormat`.

Indicates whether the format in which file revisions are stored. StarTeam 2006 uses only Native-II format.

Archive Path

Values: `text`.

Internal Identifier: `ArchivePath`.

The path to the PVCS archive or VSS project containing a file.

Archive Name (Advanced)

Values: 32–digit hex string representing the MD5 value of the file.

Internal Identifier: `STArchiveName`.

For a file stored in a Native-II Vault, indicates the name of the file that stores the tip revision. This name is the MD5 value of that file revision's content, converted to a 32–digit hex string.

Archive Type

Values: `Native`, `PVCS`, `VSS`.

Internal Identifier: `Type`.

Indicates whether a file is stored as a StarTeam (Native), PVCS, or VSS file.

Branch On Change (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `BranchOnChange`.

Indicates whether the item will branch when it changes.

The value is `No` if the item's behavior is not set to **Branch On Change**. Reasons for this may be:

- The item is in the root or a reference view and the **Branch On Change** feature is disabled.
- The item is in a branching view but has already branched as a result of a change, which, in turn, results in the **Branch On Change** feature becoming disabled.
- The item is in a branching view, but its behavior currently does not permit it to branch on change. This means that modifications are checked into the parent view.

Note: If the value is `No`, the value of the **Branch State** explains the `No`.

Branch State (Advanced)

Values: `Branched`, `Not Branched`, `Root`.

Internal Identifier: `BranchState`.

Indicates whether an item has branched in the child view, is still unbranched (and therefore is part of the parent view), or was created in the view in which it resides.

The values `Branched` and `Not Branched` apply to items in branching views. The value `Root` applies to items created in the view in which the item currently resides.

If the view is a reference view, it reflects the state of the item in the reference view’s parent.

Comment

Values: `text`.

Internal Identifier: `Comment`.

The initial 2000 characters provided as the reason for changing an item's properties or contents are stored in the **Short Comment** field. The **Comment** field stores those 2000 characters and any additional text. Changing an item's properties causes the application to create a new revision.

Note: To include a **Link** comment, the **Comment** field is the value to use in an HTML report.

CommentID (Advanced)

Values: `number`.

Internal Identifier: `CommentID`.

The ID number assigned to the revision comment. Displays `-1` if no revision comment was supplied.

Compression Level

Values: `Default`, `Maximize Compression`, `Maximize Speed`, `None`.

Internal Identifier: `Compression`.

Indicates a file's level of compression.

Default
A compromise between **Maximize Compression** and **Maximize Speed**.

Maximize Compression
The densest possible compression of file revisions. Used to save space on the server.

Maximize Speed
The fastest possible compression of file revisions. Used to improve server performance.

None
No compression.

Configuration Time

Values: `date/time`.

Internal Identifier: `ConfigurationTime`.

Indicates the time to which an item is configured. If you configure an item to a specific time, this field contains that time. If you configure an item to a label or promotion state, this field shows either the time at which the label was created, or the time at which the label associated with the promotion state was created.

Content Revision

Values: `number`.

Internal Identifier: `ContentVersion`.

The number of times a file has been checked in. If the file is in a child view, it includes all the content revisions from the parent view in this number. Each revision appears in the file’s history.

Created By

Values: list of users, `<None>`.

Internal Identifier: `CreatedUserID`.

The name of the user who created the first revision in the view. This is either the user who added the item to the project, or the user who checked in the revision that branched.

Created Time

Values: `date/time`.

Internal Identifier: `CreatedTime`

The time at which the first revision in the view was created.

Deleted By

Values: list of users, `<None>`.

Internal Identifier: `DeletedUserID`.

The name of the user who deleted the item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Deleted Time

Values: `date/time`.

Internal Identifier: `DeletedTime`.

The time at which an item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Description

Values: `text`.

Internal Identifier: `Description`.

The description provided for an item at the time it was added to the view, including any later edits to it.

Dot Notation

Values: `text`.

Internal Identifier: `DotNotation`.

The branch revision number, for example, `1.2.1.0`.

End Modified Time (Advanced)

Values: `date/time`.

Internal Identifier: `EndModifiedTime`.

The date and time at which a revision ceased to be the tip revision. Although this field can be displayed in the upper pane, its value is always blank. This is because, at any given configuration time, the item is still the tip revision.

EOL Character

Values: numeric representation of ANSI character.

Internal Identifier: `EOL`.

For internal use only. This field is primarily used to determine an ANSI character to use when breaking up lines within files for delta storage with Native-1 Vaults.

Executable

Values: `No`, `Yes`.

Internal Identifier: `Executable`.

Indicates whether the executable bit should be set for a UNIX file.

Extension

Values: `text`.

Internal Identifier: `Extension`.

Displays the extension of the file. This field is client-calculated.

File Time Stamp at Check-In

Values: `date/time`.

Internal Identifier: `Modified`.

The file’s time stamp at the time it was last checked in.

File Type

Values: `ASCII`, `Binary`, `Unicode`.

Internal Identifier: `Charset`.

Indicates whether the file is an ASCII (text), binary, or Unicode.

Flag

Values: `No`, `Yes`.

Internal Identifier: `Flag`.

Marks or bookmarks files in the upper pane on your workstation. This is a client-calculated field.

Flag User List (Advanced)

Values: text displayed as a list of user names. For example: `[Greg,Sam]` indicates user names.

Internal Identifier: `FlagUserList`.

Can be used in queries. Identifies users who have set flags on a given item.

Folder

Values: `text`.

Internal Identifier: `Folder`.

The name of the folder that stores the item. This is a client-calculated field.

Folder Path

Values: `text`.

Internal Identifier: `Folder Path` (contains spaces).

The path to the folder. This is not the path to the working folder.

Hive ID (Advanced)

Values: number assigned by the StarTeam Server.

Internal Identifier: `HiveID`.

Indicates the ID number of the hive that stores the tip revision for a file stored in a Native-II Vault.

Item Deleted By

Values: list of users, `None`.

Internal Identifier: `ItemDeletedUserID`.

The name of the user who deleted this item. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Item Deleted Time

Values: `date/time`.

Internal Identifier: `ItemDeltedTime`.

The time at which the item was deleted. Because deleted items do not appear in the list, this information is unavailable to users. Internal Use Only.

Local Name

Values: `text`.

Internal Identifier: `LocalName`.

Name of the working file. This is a client–calculated field.

Local Path

Values: `text`.

Internal Identifier: `LocalPath`.

Path name to the folder/file. This is a client-calculated field.

Locked By

Values: list of users, <`None`>.

Internal Identifier: `ExclusiveLocker`.

The name of the user who has exclusively locked a folder.

MD5 Checksum

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. The StarTeam client displays only significant zeroes so the `05` and `0A` would become just `5` and `A`, and `A–F` as `a–f`.

Internal Identifier: `MD5`.

Cannot be used in queries. The MD5 checksum for the tip revision.

Modified By

Values: list of users, `<None>`.

Internal Identifier: `ModifiedUserID`.

The name of the user who last modified the item.

Modified Time

Values: `date/time`.

Internal Identifier: `ModifiedTime`.

The time at which an item was last modified. The item may have been checked in or had its properties changed. For folders, this has nothing to do with the working folder. Use **Local Time Stamp** for the time a working folder was last modified.

My Lock

Values: `Exclusively Locked By Me`, `Non-exclusively Locked By Me`, `Not Locked By Me`.

Internal Identifier: `MyLock`.

Indicates whether the current user has the item locked and, if so, whether that lock is exclusive or not. This is a client-calculated field.

Name

Values: `text`.

Internal Identifier: `Name`.

Displays the name of the item.

New Revision Comment (Advanced)

Values: `text`.

Internal Identifier: `NewRevisionComment`.

Internal use only. The client uses this value during the item update process. The field always appears empty if added to the upper pane. This is a client-calculated field.

Non-Exclusive Lockers

Values: `text`.

Internal Identifier: `NonExclusiveLockers`.

The names of the users who have locked the folder non-exclusively.

Object ID

Values: `number`.

Internal Identifier: `ID`.

Each item is assigned an object ID when it is added to a view. For applicable items, when it is branched in a child view, it is assigned another object ID. The original ID belongs to the folder in the parent view.

Parent Branch Revision (Advanced)

Values: `number`.

Internal Identifier: `ParentRevision`.

The last digit in the branch revision number before an item branched. For example, if this number is `7`, the branch revision was `1.7` at the time the item branched (becoming `1.7.1.0`, as seen in the item's history). This number is `-1` if an item was not inherited from the parent view.

Parent ID (Advanced)

Values: `number`.

Internal Identifier: `ParentID`.

The object ID of an item in the parent view. The Parent ID is `-1` if this view has no parent view.

Parent Revision (Advanced)

Values: `number`.

Internal Identifier: `PathRevision`.

The revision number at which an item branched. For example, if this number is `8`, this item's revision number in the parent view was `8` at the time the item branched. The history should show that revision `9` in the first revision in the current view. This number is `0` if this item was not inherited from the parent view.

Path

Values: `text`.

Internal Identifier: `Path`

The path to an item's working folder. This is a client-calculated field.

Project ID (Advanced)

Values: `number`.

Internal Identifier: `ProjectID`.

The ID number assigned to a project. Within a server configuration, projects are assigned ID numbers in the order in which they are created. The first project has ID `0`.

PVCS Revision (Advance)

Values: `text`.

Internal Identifier: `PVCSRev`.

The file's revision number in PVCS's dot notation.

Read Only (Advanced)

Values: `No`, `Yes`.

Internal Identifier: `ReadOnly`.

Indicates whether the item's configuration is read-only (as in a rollback configuration of a view)/its behavior does not allow it to branch on modification. For folders, do not confuse a read-only configuration (an application issue) with a read-only folder (an operating system issue). A read-only folder cannot be edited and saved to disk. A folder whose configuration is read-only can be edited and saved to disk; it just cannot be checked in.

Revision

Values: `number`.

Internal Identifier: `ViewVersion`.

The number of times a file has been checked in or had its properties changed. If the file is in a child view, it includes all the revisions from the parent view in this number. This is a client-calculated field.

Revision Flags (Advanced)

Values: `0`.

Internal Identifier: `RevisionFlags`.

Internal use only.

Revision on Disk

Values: `number`.

Internal Identifier: `SyncPathVersion`.

The number of the revision that is currently in the working folder on your workstation. The application displays no number if the file’s status is **Missing**. This is a client-calculated field.

Share State

Values: `DerivedShare`, `Not Shared`, `Root Share`.

Internal Identifier: `ShareState`

Indicates whether this item is shared. `Not Shared` means that the item is not shared. `Root Share` means that the item is shared and this item is the original (or root) reference. `DerivedShare` means that the item is shared, but this item is not the original (or root) reference.

Short Comment

Values: `text`.

Internal Identifier: `ShortComment`.

Stores the initial 2000 characters provided as the reason for changing an item’s properties or contents. Additional text is stored in the **Comment** field.

Size

Values: `number`.

Internal Identifier: `FileSize`.

The tip revision’s size in bytes.

Status

Values: `Current`, `Deleted on Disk`, `Deleted on Server`, `Merge`, `Missing`, `Modified`, `Not In View`, `Out Of Date`, `Unknown`.

Internal Identifier: `Status`.

Indicates the relationship between the copy of an item in your working folder and the tip revision in the repository.

Storage Type (Advanced)

Obsolete.

Sync Branch Version

Values: `number`.

Internal Identifier: `SyncObjectVersion`.

A field used to determine status. The last number of the branch revision that was most recently checked out to the working folder. This is a client-calculated field.

Sync Content Version

Values: `number`.

Internal Identifier: `SyncContentVersion`.

A field used to determine status. The revision checked out as the working file or, if the file needs to be merged, a number higher than that. This is a client-calculated field.

Sync Known

Values: `No`, `Yes`.

Internal Identifier: `SyncKnown`.

A field used to determine status. Indicates whether the server knows the working file’s relationship to the tip revision. This is a client-calculated field.

Sync Local Size

Values: `number`.

Internal Identifier: `SyncSize`.

A field used to determine status. The size of the working file in bytes. This is a client-calculated field.

Sync Local Time Stamp

Values: `date/time`.

Internal Identifier: `SyncTime`.

A field used to determine status. The time stamp for the working file. This is a client-calculated field.

Sync MD5

Values: `byte array`. Displayed as a bracketed series of numbers in hex format. The StarTeam client displays only significant zeroes, so `08`, `0B`, and `06` would become just `8`, `B`, and `6`, and `A–F` as `a–f`.

Internal Identifier: `SyncMD5`.

Can not be used in queries. A field used to determine status. The MD5 checksum of the working file. This is a client-calculated field.

Sync On Path To Root

Values: `No`, `Yes`.

Internal Identifier: `SyncOnPathToRoot`.

A field used to determine status. When the working file is not based on the tip revision, this field indicates whether the server knows the relationship between the two. A `Yes` value in this field means that the working file needs to be merged or is out of date. A `No` value means that the relationship cannot be determined. This is a client-calculated field.

Vault Branch Version (Advanced)

Values: `number`.

Internal Identifier: `VaultVersion`.

The number of times a file has been checked in from the current view.

Version (Advanced)

Values: `number`.

Internal Identifier: `RevisionNumber`.

The last number in the branch revision number. For example, if the branch revision number is `1.3.1.2`, the version is `2`.

View

Values: list of views, `<None>`.

Internal Identifier: `ViewID`.

The name of the view in which the item last branched. For example, if an item is inherited from a parent view but is branched in a child view, the value of this field in the child view changes from the name of the parent view to the name of the child view for the revision that branched and subsequent revisions in the child view.

Working File Executable

Values: text.

Internal Identifier: `No`, `Yes`.

Indicates whether the working file is executable. This is a client-calculated field.

Working File Exists

Values: `No`, `Yes`.

Internal Identifier: `LocalFileExists`.

	Indicates whether a copy of a file is in its working folder. This is a client-calculated field.
Working File Size	Values: `number`. Internal Identifier: `LocalSize`. The size of the working file. This is a client-calculated field.
Working File Time Stamp	Values: `date/time`. Internal Identifier: `LocalTimestamp`. The time stamp of the working file. This is a client-calculated field.

Check-in and Check-out Operations

One of the main functions of a source control management application is to place files into a project under version control. After it is under version control, team members can check files out, revise them, and check in new revisions. The application preserves historical information about each file revision. And because of its linking capabilities, you can link a file revision to other items that affected the file or a particular revision of it. A number of other operations can be performed on files, such as moving a file or changing its branching behavior. This section specifically addresses the check-in and check-out operations.

Check-in and Check-out Overview

To place a file under version control, it must be added to a folder in a project view, which stores a copy of the file in the repository. After the file has been added to the repository, you and other members of your team can check it out, revise it, and check in new revisions, while the application maintains information on all revisions of the file. When checking out a file revision, you should verify that you have the tip or latest version of the file. Doing this ensures that the file you see contains the latest changes.

When you check in a file from a working folder on your computer, the application stores it by the MD5 value of its contents. If the file is identified as one that compresses well, it is compressed and placed in the hive's archive with a `.gz` extension. Otherwise, the uncompressed version is placed in the hive's archive.

Note: Workspace change packages are created upon the initial check-in of a file. Details about the workspace change package are displayed in the **Change** tab.

When you check out a file, the application copies the requested revision of that file to the appropriate working folder. If a copy of that file is already in the working folder, it is overwritten unless the working file appears to be more recent. In that case, you are prompted to confirm the check out.

You can perform check-out operations on more than one file at a time. For example, you can select files across multiple child folders using the All Descendants button, or you can check out all the files in the selected folder and its descendant folders using the **Check Out All** item on the **File** or context menu. This selection is equivalent to selecting **All Descendants**, all the files in the upper pane, and **Check Out**. When you use the **Check Out All** command, a confirmation dialog appears, regardless of your personal options settings.

Check-in and Check-out Recommendations

Every time you check a file revision out, its contents are copied to a working folder. Checking out a revision also ensures that you have the tip or a specific revision to work on. For example, you may need a team member’s most recent changes to a file, or you may have deleted the working file from your hard drive and now need another copy.

Below are some recommendations about using files that are under version control:

- To let other team members know that you intend to make changes to a file, change the lock status to exclusive as part of the check-out process.
- As part of the check-in process, you can notify others both that you are finished making your changes to the file and that it is available for them to check out by removing the lock status.
- If you intend to continue making changes to the file but still want to check it in for backup purposes, keep the file locked.
- If two team members change the same text file simultaneously or if one member changes an outdated file, you can use the merge option to combine the changes in these files so no work is lost. In such cases, the application gives the file a Merge status.
- To prevent yourself from changing a file that you have not locked, select the **Mark Unlocked Working Files Read-only** personal option. Then, if you check out a file that you have not locked, the working copy becomes read-only.

About the File List Display

The files that appear in the upper pane depend upon the following factors:

- The application folder you select from the folder hierarchy in the left pane.
- The filter you select from the **Filter** drop-down list box above the upper pane.
- The depth you specify with the **All Descendants** button or by selecting **All Descendants** from the **File** menu. When this button is not selected, the application displays information for the selected folder only. When this button is selected, the application displays information for the selected folder, its children, its children's children, and so on.
- The application folder you select from the folder hierarchy in the left pane.
- The sorting, rearranging, and querying you perform on the list. The application sorts files in alphanumeric order, regardless of case.
- The files you have excluded from the display by using the folder's current or inherited exclude list.

Note: StarTeam recognizes the operating systems where the case-sensitivity of file names is important and handles files accordingly.

Achieving Consistent Check-ins and Check-outs

Developers can use various StarTeam features to allow or avoid conflicts with other developers on the same files (in the same view). In a low-contention environment, developers can check-out files without locks, modify them, "refresh" to identify and resolve merge conflicts, and then check in modified files. All check-ins in StarTeam are atomic. If more than one file is checked in as the result of a single transaction (for example, in a change package from a View Compare/Merge session) the files and their associated process items are updated in a single action. If for some reason, the check-in fails, none of the files are checked in, and the status of the associated process items is not updated. In general, you can achieve consistent check-ins and check-outs by doing the following:

- Exclusively lock files before checking them in or out and unlock files after a successfully checking them in or out; or
- Temporarily change your view configuration to a known "stable" point

In higher contention environments, developers may want more assurance of getting consistent sets of files during check-out, that is, avoiding files that are a partial set of someone else's check-in. The easiest way to achieve this need is "by convention". Each developer exclusively locks all files before checking them in, and unlocks them when they are "complete", either at check-in or soon thereafter. Correspondingly, each developer exclusively locks all files before checking them out, and unlocks them when complete. If a developer cannot get all the locks at once, they are potentially about to interfere with another developer, so they unlock files they currently have locked, wait a bit (probably talk with the developer they are conflicting with), and then try again. One implication of this "by convention" approach is that a developer could be blocked while waiting for another developer to finish-up.

A more formal way to enforce consistent check-outs is to use "view configuration". To ensure consistent check-outs without locks, a developer can temporarily change their view configuration to a known "stable" point. In some organizations, a nightly build process creates a view label when the server is not used or lightly used. To temporarily change the view configuration from the StarTeam client, select **View** > **Select Configuration** > **Labeled Configuration** and choose the latest build label. (Alternatively, choose a timestamp or promotion state.) The view will switch to show the item states at the selected time, and “consistent” check-outs can be performed from there.

Note: The view configuration change is only at the client – the underlying “real” view is not modified. Also, note that “rolled-back” views are read-only: they must be reset to the “current” configuration before new/modified files can be checked-in. An implication to be aware of with this approach is that the time to switch the configuration can take a few seconds to a few minutes on very large views.

Checking In Files

1. On the **File** tab in the upper pane, select one or more files.
2. Choose **File** > **Check In** to open the **Check In** dialog box.
3. Type a generic reason for the check-in comment, or check **Prompt for a comment (check-in reason) for each file** to open a separate **File Description** dialog box for each file.
4. Optionally, click **Compare** to compare the file you are checking in with the tip revision of the file in the repository. If differences exist, the **File Compare Merge** window opens showing the file differences.
5. Select a **Lock status** option:

Unlocked	Releases your lock on the files after check-in.
Exclusive	Indicates that you intend to make further changes to the files.
Non-exclusive	Indicates that you are working on the files and may possibly make changes.
Keep current	Retains the current lock status.

6. Continue completing the **Check In** dialog box options as follows.
   - Optionally, check **Force check-in** to check in files regardless of their status.
   - Optionally, check **Delete working files** to remove the selected files from your working folder after they are checked in.
   - (Required if process rules are enforced) Check **Link and pin process item** to link the new files to process items. To use a process item besides the active process item, click **Select** and use the **Select Process Item** dialog box to change the process item.
   - If work on the active process item is now complete, check **Mark selected process item as fixed/finished/complete**.
   - To make changes to the selected process item's properties during the check-in process, check **Show property editor for selected process item**.
   - Optionally, select a **Revision label** from the list, or create a new revision label by typing its name. Existing labels are listed in reverse chronological order, based on the time at which they were created, unless the **Sort View Labels By Name** option has been selected in the **Personal Options** dialog box.
7. Optionally, click **Advanced** to open the **Advanced Options** dialog box.
   - Check **Set EOL check-out format (for text files only)** to control the EOL character stored with the files. The default setting is based on the **EOL** setting in the **File Properties** dialog box.
   - Select an appropriate check-out **File Encoding** from the list.
   - Select the file type of file you have selected: `ASCII`, `Binary`, or `Unicode`.
   - Click **Show Change Requests** to review the change requests linked to the files you are checking in.
   - Click **OK**.
8. Click **OK**.

Note: You can also use the **Check In** or **Check In And Unlock** buttons on the toolbar to check in files without using the **Check In** dialog box. If process rules are required, the **Check In** dialog box will open automatically.

Editing Check-in Comments

Normally you enter comments during the check-in process describing the changes you are checking in. You can also add or edit check-in comments after files have been checked in.

To add or edit a check-comment after a file has been checked in

1. Select the file in the upper pane on the **File** tab, and click the **History** tab or the **Label** tab.
2. Select the revision for which you want to add or edit the comment in the **History** or **Label** pane.
3. Right-click the selected revision and choose **Edit Comment** to open the **Edit Comment** dialog box.
4. Type the text you want for the comment and click **OK**.

   Comment fields allow up to 30,000 characters.

Checking Out Files

1. On the **File** tab in the upper pane, select one or more files.

   Note: You can check out files with a status of **Current**, **Out Of Date**, or **Missing**. You can also check out files with a status of **Modified**, however, you will be warned that continuing the check-out will overwrite files in your local working folder with the tip revision in StarTeam.
2. Choose **File** > **Check Out** or **File** > **Check Out All** to open the **Check Out** dialog box.
3. Optionally, check **Force Check Out** to overwrite any files with the same name in your working folder, even if they are more recent.
4. Select one of the following options in the **Reference By** group for the files you wish to check out:

   Current Revision The most current (tip) revision.

   Label A specific file revision. The existing view and revision labels are listed in reverse chronological order based on the time at which they were created. The view labels precede the revision labels in the list.

   Promotion State A specific promotion state.

   As Of The revision that was the tip revision at the specified date and time. Click the **Date/ Year** button to use the calendar, and specify the time by typing in the time or using the spin boxes.
5. Select a **Lock status** option:

   Unlocked Releases your lock on the files after check-in.

   Exclusive Indicates that you intend to make further changes to the files.

   Non-exclusive Indicates that you are working on the files and may possibly make changes.

   Keep current Retains the current lock status.
6. Optionally, click **Advanced** to open the **Advanced Options** dialog box.
   a) Select **Default working file location**, or if you want to check the files out to a folder other than your designated working folder, select **Other** in the **Check Out Location** group box . If you select **Other**, browse for the folder name.
   b) In the EOL Conversion area, click **None** or one of the other radio buttons to change your current EOL conversion setting for checking out text-based files. For Windows, the EOL marker is **CR-LF** (carriage return/line feed); for UNIX, it is **LF** (line feed); for Macintosh operating systems, it is **CR**

(carriage return). EOL settings on this dialog override the default setting you selected on the **File** tab of the **Personal Options** dialog box.

c) Select an appropriate **File Encoding** from the drop-down list to support keyword expansion for non-English code pages.

d) Click **Close** to return to the **Check Out** dialog box.

7. Click **OK**.

Note: You can also use the **Check Out** button on the toolbar to check out files without using the **Check Out** dialog box. If process rules are required, the **Check Out** dialog box will open automatically.

Monitoring Check-out Statistics using the 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.

1. On the **File** tab in the upper pane, select one or more files.
2. Choose **File** > **Check Out** or **File** > **Check Out All** to open the **Check Out** dialog box.
3. Check **Show check-out statistics**.
4. Select any other option settings that are appropriate to your check-out operation, and click **OK**.

   After the check-out operation completes, the **Check-Out Statistics** dialog box opens displaying the elapsed time, total number of files, total number of bytes, and whether any of the files were failed or skipped during the check-out.
5. Click **OK**.

Effects of Status on Check-ins and Check-outs

During the file check-out process, the application copies a file revision from the repository to a working folder. Checking in a file places a new revision in the repository. In many cases, the status of a file affects the check-in or check-out process.

Status	Check-in	Check-out
Current	No considerations.	No considerations.
Deleted on Disk	Not applicable. If a file has a Deleted on Disk status, it is not in your working folder, so it cannot be checked in.	No considerations. If a file has a Deleted on Disk status, you are asked if you want to check it out when you open it. You can also check it out manually.
Deleted on Server	Not applicable. A file with the Deleted on Server status cannot be checked in. You can add it to the project with the **Add Files** command.	Not applicable. A file with the Deleted on Server status is not in the project view, so it cannot be checked out.
Ignore	Not applicable Files with the `Ignore` status can be checked in. However, marking them as `Ignore` indicates that you do not want to work with the file.	Not applicable Files with the `Ignore` status can be checked out. However, marking them as `Ignore` indicates that you do not want to work with the file.

Status	Check-in	Check-out
Merge	Starts File Compare/Merge unless you force the check-in. The Merge status means that someone else has checked in this file since your last check-out. You do not have their changes in your working file and someone’s changes will	Your changes will be lost if you check out this file. You may be able to merge the tip revision and your working file using the application in which this file was created, for example, Word for Windows. If the file is a text file, try a check-in operation.
Missing	Not applicable. If a file has the Missing status, it is not in your working folder so there is nothing to check in.	No considerations. If a file has the Missing status, you are asked if you want to check it out when you open it. You can check it out manually, too.
Modified	No considerations. Unless someone else has the file locked, you can check in the file.	No considerations. Unless someone else has the file locked, you can check out the file.
Modified on Disk and Deleted on Server	Not applicable. A file with the Modified on Disk and Deleted on Server status cannot be checked in. You can add it to the project with the **Add Files** command.	Not applicable. A file with the Modified on Disk and Deleted on Server status is not in the project view, so it cannot be checked out.
Not in View	Not applicable. A file with the Not in View status cannot be checked in. You can add it to the project with the **Add Files** command.	Not applicable. A file with the Not in View status is not in the repository, so there is nothing to check out.
Out of Date	Not allowed unless you force the check-in. Checking in an Out Of Date file means that the tip revision no longer has the changes made to the file since the time your working copy became Out Of Date.	No considerations. Checking out an Out Of Date file makes your working file Current.
Unknown	Not allowed unless you force the check-in. If the file’s status is Unknown, the consequences of this action are also unknown. Your working file becomes the tip revision in the repository. Use Update Status with an MD5 checksum to see if the file can be identified. You might want to compare your working file to the tip revision if this is not successful.	Allowed if you merge the file with the tip revision. However, because the very first revision is used as the ancestor file for this merge, it is likely that many, many things appear to have changed or be in conflict. You may prefer to force a check-out (or force a check-in). If the file’s status is Unknown, the consequences of this action are also unknown. Your working file is overwritten by the tip revision in the repository. Use Update Status with an MD5 checksum to see if the file can be identified. You might want to compare your working file to the tip revision if this is not successful.

Checking Out Historical Versions of Files

You can easily check out a previous revision of a file using the **History** pane. You will have a choice whether to check out the file to the current working folder, which would overwrite the current file, or to check out to a different location.

1. On the **File** tab in the upper pane, select one or more files.
2. Click the **History** tab or the **Label** tab in the lower pane, and select the revision to check out.
3. Right-click the selected file and choose **Check Out** or **Check Out To**.

   This opens the **Check Out** dialog box.
4. Use the **Check Out** dialog box to check out the file as described in the procedure for checking out files.

Atomic Check-ins

All check-ins in StarTeam are atomic. Whenever more than one file is checked in as the result of a single transaction all of the files, and their associated process items, are updated in a single action. If for some reason, the check-in fails, none of the files are checked in, and the status of the associated process items is not updated.

For example, suppose `User A` selects to check in all modified files in a StarTeam folder, but one of the selected files is locked by `User B`. Because of the locked file, none of the files are checked in (and none of the process items are updated as fixed) and `User A` is notified that none of the files were checked in because one of the files was locked by `User B`.

EOL Conversion Handling Overview

StarTeam provides support for fixed EOL conversion files. For example, files can be checked out in `LF` format on every platform, regardless of specific options. Also, **Update Status** works for all text files once EOL Format is defined, regardless of what EOL format was used when they were checked-out. For compatibility with older releases of the Cross-Platform Client, if check-out "EOL conversion" is not requested, and EOL Format is `Undefined`, files are still checked out with the EOL conversion with which they were added to the StarTeam Server.

Note: The default for automatic EOL conversion for check-out operations is “checked” if the user does not have that option defined already.

EOL Format Property

The EOL Format Property displays as **EOL Character** in the Cross-Platform Client **Items** pane.

The EOL Format property is only meaningful for text files during the check-out operation.

By default, the SDK will compute the EOL Format when a new text file is added or a new revision is checked in for a text file whose EOL Format is `Undefined`, the file's EOL convention matches the platform default, or EOL Format is set to `Client Defined`. Otherwise, EOL Format is set to the convention found: `Fixed LF`, `Fixed CR`, or `Fixed CRLF` :

- The user can change **EOL Format** to any value (other than `Undefined`) at any time.
- Regardless of their **EOL Format** setting, text files always use a canonical (`CRLF`) format in the vault.

The **EOL Format** property can be manually set in the Cross-Platform Client in the **Add/Check-in** dialog boxes, and the **File Properties** dialog box. The Cross-Platform Client **EOL conversion for add/check-in** options are not available.

EOL Property Values

The EOL property value is displayed as **EOL Character** in the Cross-Platform Client **Items** pane.

The EOL Format property can be set in the Cross-Platform Client in the **Add/Check-in** and **File Properties** dialog boxes. If selected in the **Add/Check-in** dialog box, StarTeam uses the settings specified in the **File Properties** dialog box.

The EOL Property values are:

`Undefined` (null in the SDK).

`Client Defined` Causes workstation default or per-checkout EOL conversion option to be used.

`Fixed CR`, `Fixed LF`, `Fixed CRLF` Causes this EOL format to be used always; the workstation/check-out conversion option is ignored.

Controlling EOL Characters

You can specify which EOL and path case sensitivity settings are active for file check-out operations that you perform.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Select the **File** tab.
3. Select or clear **Automatic EOL Conversion On Check-out**.
4. Click the appropriate radio button to specify whether you are working on Windows, Unix, or Mac.

Process Items and Process Rules

This section discusses the use of process items and process rules.

Process Items

Modern development practices require increased control over the entire development process. StarTeam enables developers to follow a defined development process, one that ensures that all file content changes be linked to either a change request, requirement, task, or custom component. Items used in this way are known as *process items*.

Specifically, a process item is a change request, requirement, task, or custom component that is specified by the user as the reason for making a given set of changes. Process items are supported by the **Add Files** and **Check In** dialog boxes. As a result, source code and content are modified only to meet clearly defined and approved objectives, as expressed in the process item.

Out-of-view Process Items

Historically, StarTeam has supported the selection of a process item from only within the current view. This functionality is useful in many processes, but it does not support a process where change requests, tasks, requirements, or custom components live in a different view than the source code files.

To support out-of-view process, StarTeam now enables you to choose a valid process item for file add or check-in operation from any view on the same server as the files being committed. You can choose an item selected in the **Items** pane as the active process item for the current view, an open view on the same server, or a different view on the same server.

Also, the **Active Process Item** toolbar button contains a list enabling you to select the active process item from any opened view.

Process Item Selection

StarTeam currently allows a process item to be selected as the active process item, which results in that process item being used by default in the **File Add** and **File Checkin** dialog boxes. The **File Add** and **File**

Checkin dialog boxes also allow you to change the active process item prior to adding or checking in the files.

Process Items and Workspace Change Packages

Process items act like lightweight change containers. Using process items enables you to link and track changes made to your files, even when you and other members of your development team are not required to use process rules. They provide traceability, allowing you to trace file changes to their purpose or context. They also provide a way to identify file revisions for a specific change request, task, or requirement so that, for example, you can attach those revisions to a view or revision label.

Note: The creation of process tasks and the *Enhanced Process Model* has been replaced by workspace (check-in) change packages. Files linked to a process item can now be viewed in the details of the **Change** tab or view.

Workspace (check-in) change packages are created when the following actions are performed on files and folders:

- Add and check-in
- Move
- Rename
- Delete

Note: Workspace (check-in) change packages are not the same as the change packages feature in View Compare/Merge.

When files are committed using a process item, the process item is linked via a trace to a workspace (or check-in) change package representing the atomic commit of files. Additionally, if an active process item is available in the StarTeam Cross-Platform Client at the time of a move or delete of files, then a workspace change package is created for the transaction and a trace is created from the active process item to the workspace change package.

Specifically:

- The process item can be a change request, a task, a requirement, or other custom component defined for **Process Rules** in the **Project Properties**.
- The process item can live in any view. It does not necessarily have to reside in the same view where the changes are being performed.

The StarTeam administrator can enforce the use of process items for a project by establishing process rules. You define process rules in the Project Properties dialog box. Process rules specify that process items must be used when checking in files, and they establish which type of items can be used as process items.

When process rules are enforced, you must link and pin all files you add or check in to a process item. If process rules are not enforced, you can still take advantage of the linking and tracking made possible with process items. As you add files or check them in, you indicate that the new file revisions are to be linked and pinned to a specific process item. You do this by selecting a change request, requirement, task, or custom component as the process item for the operation. At the same time you can mark the change request as fixed, the requirement as complete, or the task as finished.

Note: If there is an active process item available, the **Check In** dialog box automatically fills in the **Process Item** field.

Using process items enables you to clearly distinguish the following:

- Which file revisions are related to or fix a specific change request.
- Which file revisions are related to or complete a specific requirement.
- Which file revisions are related to or finish a specific task.

Each view can have a different active process item. As you change from view to view, the process item information displayed on the status bar changes.

Process Rules

Each project has the option of enforcing the use of process items by specifying certain *process rules*. When enforced, the process rules require you to specify a specific process item (change request, requirement, or task) for file add or check-in operations within the project.

Process Rules Review

If process rules are not enforced, any change request, requirement, or task can be used as a process item, regardless of its status. However, if process rules are enforced, you may be able to select only one type of item as a process item. In addition, acceptable process items may be limited to those with specific statuses.

You can determine whether process rules are in effect for a specific project, and what those rules are, by reviewing project properties. If you do not have the access rights necessary to do this, ask your administrator what process items apply to the project and what restrictions have been placed on them.

To set process rules, you must have the access rights required to change project properties. As a rule, only team leaders and administrators have these rights. To use process items, project users must have the necessary access rights, which are the rights to:

- See and modify the types of items used as process items in the project view.
- Create and modify traces.
- View and create change packages.

Advantages of Process Rules

Establishing a system of process rules allows you to:

- Require that process items be used every time files are added or checked into the project.
- Stipulate that only certain types of items with specific statuses can be used as process items in the project.

If process rules are not enforced, linking and pinning to a process item during file add and check-in is optional, and you can select any change request, requirement, or task as a process item, regardless of its status.

Note: As a convenience, you can select a change request, requirement, task, or other custom component that has been established as a valid process item type, as the *Active Process Item* prior to adding or checking in files so the check-in automatically use that process item during check-in.

Active Process Items

The Client enables you to pre-select a process item as the active process item to use the next check-in operation. This type of process item is referred to as an *active process item*.

You can select a change request, requirement, task, or custom component as an active process item before adding or checking in files. Pre-selecting an active process item is a convenient way to save time when you know that you will be adding files or checking them in later using the designated process item. When you have a process item selected on the upper pane, making it the active process item is a simple operation. Before or during each check-in operation, make sure you select the correct *active process item*.

The active process item is the default selection when you add files or check them in. However, you can change your mind and select another appropriate item. An active process item is used until its status becomes ineligible or another process item is chosen. Additionally, if an active process item is available in the StarTeam Cross-Platform Client at the time of a move or delete of files, then a workspace change package is created for the transaction and a trace is created from the active process item to the workspace change package.

The **Status** bar displays the name of the active process item. If the active process item is from a different view and project than the current view, you can hover over the active process item shown in the status bar and see a tool tip showing the project and view as well as the process item description. The **Status** bar pre-pends the name of the active process item with the name of the project and view. To see details about a process item, you can double-click the item in the status bar to open the Properties dialog box with more information.

Note: You can only specify one item for each view as an active process item. Selecting a second active process item clears the first.

Tip: After you finish with a process item, you should choose **Clear Active Process Item** on the menu so that it cannot be accidentally reused. That removes the information from the status bar and keeps the process item from reappearing in the **File Add** or **File Checkin** dialog boxes.

Process Tasks and Enhanced Process Links

When you use enhanced process links in a project, the links for each file add or check-in operation in a given view are attached to a process task, rather than being attached directly to the process item. The process task and the process item are, in most cases, distinct. The process task is created automatically.

In the enhanced model, new process links are always added to a process task whose status is In Progress in the target view. If there is no process task in progress for a given file add or check-in operation, StarTeam creates one just before adding the first process link for that operation.

Note: A process task that is `In Progress` can itself be selected as an active process item.

Advantages

By using enhanced links, StarTeam creates the process task to track the changes for a process (such as fixing a bug or finishing a task) rather than using the process item itself.

In the enhanced linking model, StarTeam items still represent process items and add or check-in operations, and process links identify the changed files and folders. However, the process item and the changes are represented by separate items.

Specifically,

- The process item can be a change request, task, or requirement.
- The process item can live in any view. It does not necessarily have to reside in the same view where the changes are being performed.
- A process task is view specific. It is always created in the root folder in the view where the changes have been made.
- The creation of a process task for a given process item is always created automatically, and as transparently as possible. For example, if the you select a change request as a process item, StarTeam automatically creates the associated task if necessary, and automatically attaches the process links to the task instead of the change request.

By using enhanced process links, VCM always creates a new task (a process task) to represent the change package, and creates a new process link to the process task. It never modifies existing links, and therefore avoids corrupting the change package. This model makes it easier to see how a process item was used across various views in the project, no matter where the process item is referenced. It also makes it easier to support process items from another view.

Note: For projects that use the enhanced linking model, the Link tabbed pane gives you the option of displaying all links or only enhanced process links. In addition, the Link pane displays the current Item Selection and allows for navigating to child links.

How Process Tasks Work When Checking In Files

In the standard linking model, if you check in three files using a process item, the process item is linked and pinned to the tip revision of the files. With the enhanced linking model, you use process tasks for check-in and add-file operations, and the active process item is linked directly to a process task. That process task is linked and pinned to the file revisions. The process task serves as an intermediate item between the process item and the files. A process item is relevant to only one view. If you re-open and reuse a process item, StarTeam creates a new process task for it. If you use a process item repeatedly with the same process task, StarTeam creates new links for each operation.

Distinguishing Process Tasks from Other Tasks

When you select an enhanced process item, the **Link** pane displays a list of all process tasks that are relevant to this process item, regardless of which view the corresponding operation occurred in. Also, the name of the relevant view is shown in each case.

StarTeam distinguishes a process task from a standard task in the Link pane by displaying a different icon for each type of task.

You can also distinguish process tasks from standard tasks by looking at the value of the task's Usage property in the **Detail** pane.

Note: To make the Usage property visible in the **Detail** pane, you must select it as one of the Fields to show in the **Filters** > **Show Fields** dialog box.

The following are the Usage property values used for the various types of tasks:

- For process tasks created when adding and checking in files, the Usage property value is `Checkin`.
- For process task created by View Compare/Merge, the Usage property value is set to match the merge type: `Promote`, `Rebase`, or `Replicate`.
- For tasks that are not process tasks, the Usage property value is `Other`.

Creating External Links

A link is a connection between two folders, two items, or a folder and an item on the same server, or on two different servers (called *External Links*).

Creating links can be quite useful. For example, linking a file to a change request allows you to mark it as fixed when you check in the edited file. By linking files to the requirements document that the files fulfill, you can easily refer to or update the document.

Note: When you create external links between items on different server configurations, both server configurations need to be opened in the StarTeam Cross-Platform Client to be able to create or view the external links. Also, to create external links, you must have the access rights to the generic external link access rights, such as create, see, modify, delete.

1. Open both projects and servers which have items you want to link. External links will not work unless both servers are opened in StarTeam before you create the link.
2. Begin the link process by doing one of the following:
   - Select a folder from the folder tree or in the upper pane on the **Folder** tab.
   - Click a component tab in the upper pane, such as **File**, **Change Request**, **Requirement**, **Topic**, or **Task**, and select one or more items.
3. Right-click the selected item(s) on the component tab and choose **Links** > **Create Link** . The **Links** menu is also available on the component menu that corresponds with the selected component tab.

   This action changes the mouse pointer and displays it as a knotted rope.

   Note: If you initially select an item from the upper pane, you can also use the **Link** button on the toolbar. However, this button is disabled if you start the link with a folder.
4. Select the folder or item(s) for the end of the link in the project on the other StarTeam Server. This can be:
   - A StarTeam folder (if you have not already selected a folder).
   - One or more other files.

- One or more change requests or change packages.
- One or more requirements.
- One or more topics/responses.
- One or more tasks/subtasks.

To locate all items, you may need to switch to a different component tab or use the **All Descendants** button on the toolbar.

5. Click **Folder Tree** > **Links** > **Complete Links** or click the **Link** button again on the toolbar.

   This button is disabled if you are linking an item to a folder.
6. Verify that the links exist by doing one of the following:

   - Select a linked item, then click the **Link** tab on the lower pane to view the links for the item.
   - Right-click a linked folder, then choose **Properties** to display the **Folder Properties** dialog box. Click the **Link** tab to view the link. (The **Link** tab will not appear in this dialog box if you do not have access rights to view links.)

   You can also view a link by selecting either of its ends. The end you select, whether a folder or an item, is called the source. The other end of the link is called the target and is listed in the **Item Type** column on the **Link** pane.

Tip: If you change your mind about creating a link after you have started to create it, but before you have finished completing it, you can select **Links** > **Cancel Link** on the **Folder Tree** menu, the component menu, or the context menu. If you are using the **Link** button on the toolbar, press **ESC**.

Note: External links can also be created using drag-drop. With both views open, select the source item, press **CTRL + SHIFT**, then drag-drop it on the target item.

Checking Linked Files In and Out

If you are checking in a file that has one or more linked change requests, you can do this from the **Link** pane.

1. Select an item in the upper pane.
2. Click the **Link** tab in the lower pane.
3. Select one or more files in the **Link** pane.
4. Right-click the selected item in the upper pane or the selected files in the lower pane.
5. Do one of the following:

   - **Linked Files** > **Check In All**
   - **Linked Files** > **Check Out All**
6. Use the **Check In** or **Check Out** dialog box as you normally would to check files in our out.

Displaying Only Enhanced Process Links

If your project previously had enhanced process links enabled, you can choose to display only enhanced process links in the **Link** pane.

1. Click the **Link** tab in the lower pane.
2. Select the **Show enhanced links only** option at the top of the **Link** pane. When you select an item in the upper pane that has links, the **Link** pane only shows enhanced links, not standard links.

Establishing Process Rules for Projects

Establishing a system of process rules allows you to:

- Require that process items are used every time files are added or checked into the project.
- Stipulate that only certain types of items with specific statuses can be used as process items in the project.
- Enable the use of enhanced process links for the project.

Note: To set process rules, you must have the access rights required to change project properties. Usually, only team leaders and administrators have these rights. You must also verify that project users have the rights to see and modify items in the project view, to create and modify links on files and process items, and to create tasks and link to tasks if using the enhanced model.

1. Choose **Project** > **Properties** . The **Project Properties** dialog box opens.
2. Click the **Process Rules** tab.
3. Select the **Require Selection Of Process Items When Files Are Added Or Checked In** check box.
4. Select the type you want to allow for use as process items.
5. You can define the use of the type as a process item in the **Process Item Details for <Type>** section.

To permit the use of any type as a valid process item:

1. Select the desired **Type**.
2. Specify the **Active States** that are permitted to be used as a process item during commit.
3. Specify the **Closed State** that will be used to mark the process item as completed upon successful check-in.
4. Add the <Type> as a valid process item type.

Note: Some StarTeam integrations do not recognize process rules and will ignore them.

Filtering Process Tasks From Other Tasks

If you have previously enabled enhanced process links in your project so that StarTeam created process tasks, you can filter your tasks to separate the process tasks from the regular tasks. Use the **Usage** field value to determine the difference between process tasks and standard tasks. If the **Usage** value is anything other than **Other**, then it is a process task.

1. Choose **Task** > **Filters** > **Filters** . The **Filters** dialog box appears.
2. Click **New** and give the new filter a name.

   Alternatively, copy an existing filter by selecting it, clicking **Save As**, and giving it a new name. Then select the copied filter and continue with the next steps.
3. Click **Fields** in the **Filters** dialog box.
4. Move the **Usage** field from the **Available Fields** list to the **Show Fields in this Order** list and click **OK**.
5. Click **Query** to open the **Queries** dialog box, and click **New**. This opens the **Edit Query** dialog box.
6. Type a **Name** for the new query, and choose the following in the **Condition Node** section: **Field** = **Usage**, **Operator** = **Not Equal**, and **Value** = **Other**.
7. Click **Add** to add the condition to the query, and **Save** to save the query and return to the **Queries** dialog box. Your new query is now highlighted in the list of queries.
8. Click **Select** in the **Queries** dialog box use this query in your new filter. You are returned to the **Filters** dialog box, and your new filter should be highlighted.
9. Click **Save As** to save the filter.

   Notice your new filter has been added to the **Filter** list at the top of the client.

To use the filter, simply select it from the **Filter** list on the **Task** tab. Conversely, you can create a filter that displays only the standard tasks. In the query, use the condition **Usage Equals Other**.

Tip: If you use tasks on a regular basis, and not just for process tasks, add **Usage Equals Other** to existing queries so you never see process tasks when working on tasks that have been manually created, or imported from usingStarTeam Microsoft Project Integration.

Finding Files Associated with Active Process Items

When you have files associated with an active process item, you can quickly find all associated file changes by following these steps.

1. Open the pane that contains the active process item.

   You can see what item is the active process item by looking at the left side of the **Status Bar**. The second box in the **Status Bar** displays the **Active Process Item** icon, followed by the name of the item.
2. Select the active process item and click the **Change** tab in the lower pane to see all of the workspace (check-in) change packages for which the selected active process item was used.

Linking Files to Process Items

If process rules are enforced for a project, linking and pinning new file revisions to a process item is required. Otherwise, this step is optional, and you can select any change request, requirement, or task as a process item.

Promoting File Changes Into Baselines

Process rules are useful when creating baseline builds or configurations. A build is a labeled configuration that identifies the file revisions and process items that define the code and content baseline. Process rules require that each new file revision be linked to a process item, which allows the development team to promote these changes into baselines.

If process rules are not enforced, developers using the application can create baselines either by:

- Labeling an entire project view at a specific point in time.
- Associating file revisions with a revision label on check-in.

1. Start with the previous baseline (for example, check it out based on its label.)
2. Select process items for inclusion in the new baseline.
3. Label the new baseline You can use process items for tracking purposes when adding or checking in files.

View Compare/Merge (VCM)

At key milestones in a development project, most views have changes that need to be propagated to another view, and sometimes to several views. When work is complete or reaches a suitable release point in an activity or sandbox view, it needs to be merged “up” to the parent view. When a bug is fixed in the main view, it may need to be propagated “down” to one or more release views. Sometimes a fix needs to be propagated “sideways”, from one release view to a sibling release view.

StarTeam provides you with a comprehensive cross-view change management tool for propagating discrete sets of changes from one view to another called View Compare/Merge.

View Compare/Merge is available through two application interfaces:

- The Cross-Platform Client provides a graphical interface that is integrated with other CPC functionality. The graphical View Compare/Merge tool provides interactive compare/merge functions with per-item and per-folder interaction. This allows you to carefully control which items are compared, and how each difference is resolved. For example, you can select an item, folder, or group of items in a regular view window and start a View Compare/Merge session using the selection as the source scope.

- The `VCMUtility` is a command-line tool that allows you to start, resume, and commit View Compare/Merge sessions from shell scripts. In a single script, for example, you could create a View Compare/Merge session, perform a test build of the proposed changes and run other verification tasks, then commit the session if all tests complete OK.

Compared to the use of process items, which tracks file changes within a view, View Compare/Merge is the preferred mechanism for propagating and tracking changes made across views, that is, from one view to another. A common use of View Compare/Merge is to propagate the changes in one view, as tracked by a process item, to another view.

About View Compare/Merge Sessions

View Compare/Merge uses a “session” concept, which is a specific view compare/merge operation that can be started, reviewed, adjusted, verified, and then committed. When you save a View Compare/Merge session, a *change package* object is created that contains all the changes and information about the changes made during that session. Except for locks, no data is modified in the StarTeam repository until the change package for that change session is committed, at which point all changes are applied at once to the target view.

Before a change package is committed, it can be saved and restored – potentially on a different machine. This allows one user to create the change package, then transfer it to a peer for review or to verify the change package before committing it.

The Cross-Platform Client interface allows user interaction with the entire View Compare/Merge process. The `VCMUtility` provides partial or complete automation of View Compare/Merge sessions, which can be helpful when a specific kind of session is performed repeatedly. View Compare/Merge sessions are interoperable between the two interfaces. For example, you can create and save a change View Compare/Merge session using the `VCMUtility`, then restore, review, and commit the session using the Cross-Platform Client. You can also apply the same changes in the committed change package to another view using a process called “Replay”.

View Compare/Merge Session Views

Two StarTeam views are involved with every View Compare/Merge change session: a *source view* and a *target view*.

- The source view contains the changes to be propagated. You can use the tip configuration of the source view or a snapshot defined by a view label, promotion state, or timestamp. There are several ways to identify the items to be “considered” for propagation, collectively known as the source scope. If you like, the entire view can be used as the source scope, though you will usually propagate items that represent specific changes.
- The target view receives the propagated changes, so it must be able to be updated. “Receiving” the changes does not mean that the source items are merely copied to the target view as-is. View Compare/Merge compares the source view to the target view and determines if and how to propagate item differences. Propagation of a specific change can take many forms such as sharing, re-pinning, merging, and so forth. You can review and adjust what View Compare/Merge proposes before anything is committed.

Note: The source and target views must belong to the same project.

View Compare/Merge Merge Types

Every View Compare/Merge merge session has a specific *merge type* which controls the direction and semantics of the View Compare/Merge session.

Rebase A rebase session propagates changes from a parent view (the source) to an immediate child view (the target). The most common use of rebase sessions is to “catch-up” a child view with changes that occurred in the parent since the child view was created or since the last rebase was performed.

Promote A promote session is the opposite of a rebase. It propagates changes from a child view to its immediate parent. Promote sessions employ special rules that accommodate the directionality of share trees. That is, they strive to keep share trees pointing in the same direction as the project's views.

Replicate A replicate session propagates changes from a source view to any other (updateable) view in the same project. Compared to rebase and promote, replicate is the "wild card" merge type that allows changes to be propagated anywhere in the same project.

Compare-only A compare-only session only shows you the differences between views or two configurations of the same view. No changes or merge actions are performed.

Why Use View Compare/Merge?

There are a variety of reasons you might want to merge two views. A typical example would be code development in a software development company. If both maintenance and new development for a software product need to occur in parallel, a separate view is often created to store each set of source code files. If you fix some defects in the maintenance view, you may need to merge changes from that view into the new development view so that the next release of your product has the fixes.

You can compare and merge any two views if those views are in the same project. However, in general, you can merge only items that are related (that is, have a common ancestor), even though that ancestor may not be in either of the views you are comparing or merging. The exception to this is when files are matched by name.

View Compare/Merge allows you to do the following to reconcile differences between views:

- Catch up an activity or child view with new changes from its parent (`Rebase`).
- Deliver changes or a completed activity from a child view to its parent (`Promote`).
- Duplicate a fix or enhancement from one view to another view (`Replicate`).
- Merge items of one view, known as the source view, with items in a second view, called the target view. Changes are stored as updates to the target view.
- Merge two views using an earlier configuration of the source view.
- Compare and merge file contents and item properties.
- Control what is being merged by using one of several selection methods before starting the View Compare/Merge session. For example, individual items may be selected, a folder and its contents may be selected (optionally recursively), or a process item and its linked files can be selected.

View Compare/Merge always uses the source file (that is, the "merge from" file) from the last recorded merge as the common ancestor for performing three-way merges. When there is no recorded merge point, View Compare/Merge uses the most recent common ancestor. For example, if the item in the source view has the dot notation `1.9` and the item in the target view is `1.7.1.4`, the most recent common ancestor is `1.7`.

Auto-merge

The **View Compare/Merge Wizard** has an available option called **Auto-merge Files**. When this option is selected in the wizard, View Compare/Merge automatically merges files without conflicting differences at the beginning of the View Compare/Merge session.

Auto-merge does a 3-way merge. The auto-merge examines the following in the merged file:

- What is in all three files.
- What is in the child that is in neither the ancestor nor the root.
- What is in the root that is in neither the ancestor nor the child.

That means that it ignores lines that exist in both the child and ancestor, but not the root. It also ignores lines that exist in both the root and ancestor, but not the child. The former would be lines that have been deleted from the revision in the root view, and the latter would be lines that have been deleted from the revision in the child view.

It is possible that the merged file would be identical to the target file, even when the source and target files are different. For example:

Comparisons are made of an ancestor file `foo.txt` with the target (root view) `foo.txt` and with the source (child view) `foo.txt` The comparisons show that two lines in question were deleted when the ancestor revision moved to the root (target) revision, but were not deleted from the child (source) revision. Therefore these two lines were in both the ancestor and the child, but not in the root revision. In this case, the two lines in question were deleted from the root view revision. The auto-merge did not restore deleted lines.

You can perform a manual merge when you want to restore deleted lines, and you can perform an overwrite if you want the target (root) to exactly match the source (child). Even if the file is resolved with auto-merge, you can select overwrite or merge again manually.

Most Commonly-used Views

Below are the most common views you will use in view comparisons and merges:

Main View The main view is where new development eventually appears.

Activity View The activity view is a child view used for a set of related tasks such as the development of a new feature. This view generally ends when the activity is complete.

Release View The release view is a child view used to maintain a specific version of an application or module. This view generally ends when the corresponding application or module is no longer supported.

Most Common Merge Tasks

The most common tasks associated with merging views are:

- Allowing an activity view to catch up with new changes from its parent (Rebase).
- Delivering a completed activity to its parent (Promote).
- Duplicating a fix or enhancement to one or more development views (Replicate).

View Compare/Merge Terminology

This list defines common terms used in View Compare/Merge.

Action A strategy for resolving an item difference between a source view and target view. A default action may be chosen based on merge type and user-specified options, but the action may be changed by the user before it is enacted. There are many kinds of actions including ignore, share, reverse-share, delete, overwrite, merge, re-pin, and move. Combination actions are also possible for certain differences: move and merge, move and re-pin, and so on.

Activity View A branch-all variant (derived) child view whose purpose is to support a constrained-length development activity. For example, an enhancement or bug request is needed, so an activity view is created from the main view as of a specific milestone (for example, view label). Changes are made in the activity view, causing items to branch. Occasionally, changes made in the main view must be replicated to the activity view. Eventually, changes made in the activity view are "released" back to the main view, and the activity view becomes inactive.

Branch

If a new revision of an item is created in a shared location while the **Branch On Change** property is on for the item, a new branch is created for that item and is reflected in the dot notation for the tip revision. Once the item has branched, it will not branch again unless it is shared again.

Note: Only files, folders, and change requests can branch. Other item types do not branch.

In the location where an item was first created, it cannot branch. When an item is shared, it can acquire branching behavior. By default, the shared item acquires the branching behavior set by the **Branch On Change** property for the view, but this property can be reset manually for each item. If the behavior is set to **Branch On Change** , the next revision will branch, adding a new pair of integers to its dot notation.

If you compare the dot notation for two revisions, you can determine if they are on the same branch or on different branches. If the two dot notations have the same number of integers and those integers are identical (except for the last integer), then the revisions are on the same branch. A difference in the number of integers, or a difference in the value of any integer except the last, indicates that two matching items are on different branches. For example, `1.7` and `1.7.1.0` are not on the same branch. `1.7.1.0` and `1.7.2.0` are not on the same branch. `1.7` and `1.9` are on the same branch. `1.7.1.8` and `1.7.1.15` are on the same branch.

Common Ancestor

The common ancestor of two matching items is the most recent revision of the item to appear in the histories of both items, or if merge points are being used, the source revision for which the merge point has been set. For example, suppose that the root view is the source view, one of the branching child views of the root is the target view, and merge points have not been set. If the item in the source has the dot notation `1.9` and the item in the target is `1.7.1.2`, the common ancestor is `1.7` and probably was the tip revision at the time that the child view was created. View Compare/Merge uses the common ancestor of the matching items to determine if changes have been made to the source and target items.

Conflict

The properties or contents of two matching items require manual intervention in a merge situation. For file contents, conflicts arise if two matching blocks or lines of code have both been changed since the common ancestor or the most recent merge point. For properties, if a property has changed in both matching items, that property produces a conflict that must be resolved manually. If the property value of only one of the items has changed, that property does not have a conflict. If only the source property has changed, the difference for that property can be automatically resolved by overwriting the property value in the target with the property value in the source. If only the target property has changed, the difference is ignored.

Difference

Used to describe an item that is not identical in both the source view and the target view. For example, the item could be present in one view and not the other; it could be in both views but in different folders; it could be in both views but have different revisions, and so on.

Dot Notation

A set of pairs of integers separated by periods indicating the branch that each revision is on and its place on that branch.

For example, suppose you add a file to a root view. The first revision of that file has the dot notation `1.0`; the second revision has the dot notation `1.1`, and so on. Suppose you create two branching views as children of the root view, and both child views include the file `info.htm`, and the tip revision of `info.htm` is `1.7` at the time the child views were created. Both child views will show revisions `1.0` through `1.7` in the history for `info.htm`. The history also indicates that those revisions were created in the parent view. Now suppose that someone changes `info.htm` in the first child view and checks in the file. The new revision (the first revision of `info.htm` in that particular child view) will have the dot notation `1.7.1.0`. This is a new branch. To continue with the same example as before, suppose that a developer makes a different set of changes to `info.htm` in the second child view. As those changes are checked in, the dot notation of the new revision (the first revision of `info.htm` in the second child view for this file) becomes `1.7.2.0` and starts a new branch. The dot notation cannot become `1.7.1.0` because that branch has already been created in the first child view.

Floating Shared items not configured to a specific label, promotion state, or timestamp are said to be floating. For example, suppose an item with the revision `1.7` was shared into a second view and allowed to float. As the original item moves to revisions `1.8` and `1.9`, so does the shared item. When pinned, a share stops floating and stays at the revision number it is pinned to, unless it branches.

Ignore View Compare/Merge takes no action. It is always available as the action to be performed on any pair of matching items. It is sometimes the default action and sometime the only action available. For example, if an item has been deleted from the source, but its matching item is not only in use but has changed in the target since the last merge, View Compare/Merge expects that you will want to keep this item in the target view. The default action in this case is to ignore the differences between the two matching items.

Merge When this action is selected, View Compare/Merge merges the properties of the matching items and creates a new revision in the target view. The merge can be done automatically or manually. If there are conflicts, the merge must be manual. View Compare/Merge also sets a merge point for future reference.

Merge Point A merge point indicates that a merge occurred between a particular revision of an item in the source view and a particular revision of its matching item in the target view. Merge points are directional in that they denote which item was the source and which was the target. It takes precedence over the use of the common ancestor. In future comparisons and merges, if neither the source nor target item has changed since the merge point was set, View Compare/Merge takes no action. A merge point is set for any **Merge**, **Overwrite**, or **Mark As Resolved** action.

Merge Type The overall view compare/merge strategy. The possible merge types are Rebase, Promote, and Replicate.

Promote A Promote updates a parent view to reflect the changes made in one of its children. Changes in the child view usually result in branching, so changes are usually merged into the parent view. If the item exists only in the child, it can be reverse-shared into the parent. A reverse-share is a good idea because the parent view is the best location for an item that is the root of a reference tree.

Rebase A Rebase operation updates a child view to reflect changes made in its parent view. Items that have not changed in the child are rebased simply by re-pinning the item in the child view to match the current revision of the matching item in the parent.

Example of a Rebase: Suppose you create a child view based on the root view one week and wish you had waited. Several changes have gone into the root view in the past week, but little or no work has occurred in the child. A Rebase operation changes the child view so that each item is re-pinned to the current revision in the parent. If an item is new in the parent, it is shared into the child. If an item has branched in the child, re-pinning is not possible and a merge is necessary.

Release View A branch-all variant (derived) child view whose purpose is to support continuing maintenance activities for a specific product release. A release view is created from the main view as of a specific milestone such as a view label. Changes to the release view support bug fixes or minor enhancements. Some of these changes must be replicated back to the main view or other release views. Some changes made to the main view must be replicated to one or more release views.

Re-pin An item in a target view can be re-pinned if the two matching items are on the same branch. The act of re-pinning changes the common ancestor for a pair of matching items. It also moves the tip revision in the target view forward at least one revision. For example, if the item in the source view is `1.9` and the item in the target view is `1.7`, re-pinning changes the tip revision in the target view to the `1.9` revision. It is equivalent to setting the configuration time of the item forward to a time after the creation of the `1.9` revision in the source view. The common ancestor for the two items is now revision `1.9`. Re-pinning is most commonly done in a Rebase operation when the source view is the parent of the child view.

Replicate For views that are not parent/child, View Compare/Merge expects you to perform a Replicate operation. The actions available during a Replicate operation and the default action select by View Compare/Merge are the same as for a Rebase operation. However, while re-pinning may be the most used action in a Rebase operation, it may be the least used action in a Replicate operation. This is because, depending on the relationship between the views, fewer of the items will be on the same branch.

Reverse-Share When an item is reverse-shared, it is moved from the source view to the target view and then shared back into the source. The items in both views have the same data, the same number of revisions, the same dot notation, and so on. However, the reference tree changes: the item in the source view becomes the child of the item in the reference tree of the target view, reversing the previous parent/child relationship. The only differences between the original item in the source view and that item as newly shared back into the source view is its placement in the reference tree and its **Branch On Change** status. That status is on or off, depending on the default setting for **Branch On Change** in the source view. The item in the target is identical to the item that was originally in the source, although its location has changed. It has the same properties and the same place in the reference tree.

For example, when an item M is shared from folder A to B, M is simply copied from A to B, creating M^1 in B as a child share of M. In a reverse-share from A to B, M is moved from A to B and then copied (shared) back to A, creating M^1 in A as a child share of M. This technique “transfers ownership” of the item from A to B.

Share An item is shared manually by performing a share (drag-and-drop) operation. An item is shared automatically when a child view is created. Shared items can branch. When an item is shared, either manually or automatically, from one location to another, the item in the new location has the same data, the same number of revisions, the same dot notation, and so on. However, a new reference to this item appears in the reference tree for this item. The only differences between the original item and the shared item are their locations and, perhaps, their **Branch On Change** status. That status is on or off, depending on the default setting for **Branch On Change** in the receiving view. Later, the **Branch On Change** status can be changed manually. When a shared item with its **Branch On Change** status is changed, it branches and the **Branch On Change** status is disabled.

Source View A view that is compared with a target view, and whose changes are potentially merged to the target view. In some cases, the source view can use a rolled-back and/or read-only configuration.

Target Types The item types that are to be considered during a view compare/merge operation.

Target View A view that is compared with a source view and which potentially receives changes from the source view. The target view must be able to be updated, and it must be a variant (branching) view or a non-derived (blank) view.

Full Change Package Rollback

The new **Rollback** menu item in the Change Package perspective rolls back all of the files attached to the committed Change Package.

A new workspace Change Package is created and can be committed. The revision attached to the Change Package is reverse merged out of the tip content. The reverse merge process itself is manual. You are prompted to run the merge using the *Delta II* tool. When the reverse merge is complete, the updated content is stored as part of the session. When committed to the current view, you get the desired effect.

VCM Commit with Full History

Multiple revisions of a file (it's history) can be propagated to the parent view during a VCM promote session. This is typically likely if you have checked in multiple versions of a branched file in the child view, and then you ran the promote and want to carry the history of all subsequent revisions since the last promote.

Since the historical revisions in the child view may have been created by other users (than the user doing the commit), the person performing the commit must be granted impersonation rights at the server.

This feature can only be turned on at the project level using the new project option: **Merge File History during VCM commits**, and is supported on StarTeam Server versions 16.0+.

VCM Content Differences Display

The File Content differences pane for File Compare/Merge operations has been expanded to include two new panes. The first pane displays the file content differences between the Source File and the Common Ancestor picked for the merge. The second pane displays the file content differences between the Common Ancestor and the Target File.

VCM Process Item and Attached Revisions

The VCM wizard has been expended so that if a Story is used as an aggregator (rollup) of attached process items, the Story resolves to in-view Tasks created by Agile, and the in-view Tasks are used as cross project process items to files in a completely different view. A Sprint can be used for VCM as an aggregator of attached Stories. The attached Stories follow the same rules. The VCM wizard automatically changes its context to the project/view in which the files reside, and runs the VCM session for the *attached* files.

VCM Skip Multiple Revisions

The **Skip-A-Revision** feature in VCM has been modified to support skipping multiple revisions. Each selected revisions needs to be reverse merged out of the content using the merge tool *Delta aII*.

Access Rights Required for StarTeam View Compare/ Merge Users

This section explains the minimum access rights needed by View Compare/Merge users. It is recommended that project administrators manage these rights for View Compare/Merge users since they usually do more with StarTeam than create View Compare/Merge sessions.

It is assumed that your company uses all item types, which may not be the case. For example, if you do not use or do not have the Requirements component, there is no need for requirement access rights for View Compare/Merge.

Note: Before you can create access rights for View Compare/Merge Users, your server administrator needs to set up a View Compare/Merge Users group. Give your server administrator the list of users to add to this group. Once the View Compare/Merge Users group is set up on the server, you can open the **Project Access Rights** dialog box (**Project** > **Access Rights**) and set the following access rights for the View Compare/Merge Users group.

Project Access Rights

The user must be able to see the project that contains the views to be compared and/or merged.

Generic object rights	See object and its properties.

View Access Rights

The user must be able to see the view. Also, since there are options in the View Compare/Merge that allow the user to create view and/or revision labels, the user should have the rights to create both types of labels.

Generic item rights	See object and its properties.
View specific rights	Create view and revision labels.

Change Package Access Rights

Since the View Compare/Merge session creates a change package object , the user must be given appropriate access rights to change packages.

Generic item rights	Create, see, and modify properties of change packages.
Label rights	No label rights access are needed.
Link rights	Create, see, modify, and delete links.

Promotion State Access Rights

No Promotion State access rights are needed.

Child Folders Access rights

The user must be able to see folders and modify their properties because their properties may be merged during the View Compare/Merge session. For example, a folder’s name may have changed.

Generic item rights	See item and its properties. Modify properties. Delete from folder. Perform maintenance. Set and break exclusive locks.
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
Generic item container rights	Share/move folder from its parent folder. Modify folder behavior or configuration.

Additional Information About Child Folders Access Rights

- View Compare/Merge users need the `Delete from folder` access right. View Compare/Merge or the user may select the **Delete** and **Delete and reverse share** options. Since View Compare/Merge does not create folders (it only shares and reverse-shares to create new items), the `Create folders` access right is not necessary. However, users of View Compare/Merge would be very likely to need this right for their everyday work.

- There are options in the **View Compare/Merge** wizard that require the rights to set and break locks. These options are **Lock target for difference**, **Lock source for difference**, and **Break locks automatically**. Anytime a View Compare/Merge session results in a reverse share, the user needs the access rights to lock items in both the source and target views. A reverse share can occur during Promote or Rebase operations. A reverse share causes item locking in both views because the item is moved from the source to the target and shared back to the source. So, even if no locking options are chosen when using the **View Compare/Merge** wizard, the user needs the right to lock items in both views any time a reverse share occurs. A reverse share can also result in other changes to the newly shared item in the target. The item's configuration and behavior may have to be reset, and the labels that the item previously had (in the source before the move) are reapplied.
- Notice that the user needs the right to adjust both revision and view labels. Creating a revision label is not the same as attaching that label to an item. When created, a revision label is attached to nothing. View Compare/Merge performs attachments as part of the session. While view labels are attached to every item in a view as part of the label's creation process, users need to be able to adjust those labels. For example, View Compare/Merge may need to restore the labels to the child folder that the folder had before it was reverse shared.
- Reverse shares happen during promote sessions and perhaps some replicate sessions. For reverse shares, users need the rights to `Share/Move folder from its parent folder` and `Modify folder behavior or configuration`.
- The `Modify folder behavior or configuration` access right is also required when the **Fix floating child shares** option is used in the **View Compare/Merge** wizard, and when a re-pin action is selected either by View Compare/Merge or the user.

File Access Rights

Users need the same access rights for files as for child folders. In addition, they need the rights to check files in and out. Files are checked in as part of the commit portion of a View Compare/Merge session. Checking in a file requires the user have the ability to see items and properties, modify properties, see item history, and set exclusive locks. Files are checked out by users who use the View Compare/Merge session **Test Perspective**.

Generic item rights	See item and its properties. Modify properties. Delete from folder. See history. Set and break exclusive locks.
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
File specific rights	Check in and check out file.
Generic item container rights	Share/move folder from its parent folder. Modify folder behavior or configuration.

Change Request Access Rights

The access rights for change requests are the same as for child folders, and for the same reasons.

Generic item rights	See item and its properties. Modify properties. Delete from folder. Set and break exclusive locks.
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
Generic item container rights	Share/move folder from its parent folder. Modify folder behavior or configuration.

Requirement Access Rights

The access rights for requirements are the same as for child folders, and for the same reasons.

Generic item rights	See item and its properties Modify properties Delete from folder Set exclusive locks. Break exclusive locks
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
Generic item container rights	Share/move folder from its parent folder. Modify folder behavior or configuration.

Task Access Rights

The access rights for tasks are the same as for child folders and they are needed for the same reasons.

Generic item rights	See item and its properties. Modify properties. Delete from folder. Set exclusive locks. Break exclusive locks.
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
Generic item container rights	Create items. Share/move folder from its parent folder. Modify folder behavior or configuration.

Topic Access Rights

The access rights for topics are the same as for child folders and for the same reasons.

Generic item rights	See item and its properties. Modify properties. Delete from folder. Set exclusive locks. Break exclusive locks.
Label rights	Attach/Adjust view and revision labels.
Link rights	See and create links.
Generic item container rights	Share/move folder from its parent folder. Modify folder behavior or configuration.

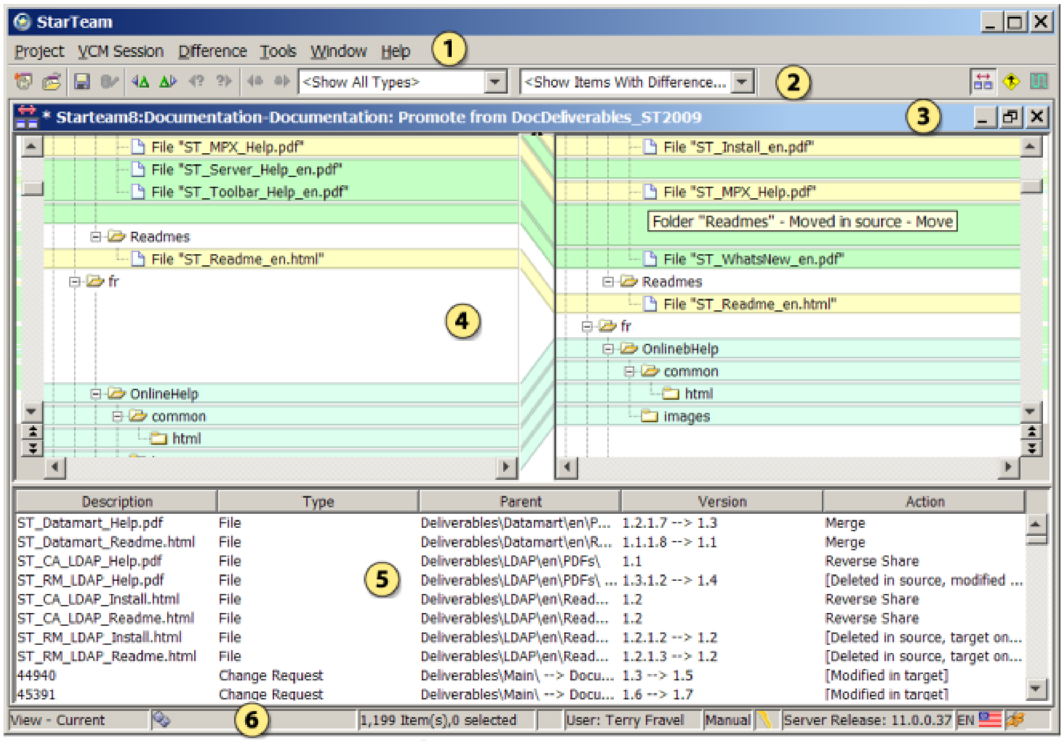
View Compare/Merge Session Perspectives

The View Compare/Merge window provides three perspectives for comparing and merging views, and for resolving item differences.

- **Compare Perspective**
- **Merge Perspective**
- **Test Perspective**

Menu items and toolbar buttons at the top of the View Compare/Merge session enable you to change perspectives. The **Title** bar of the View Compare/Merge session window displays an icon on the left indicating which perspective is being used.

Compare Perspective


StarTeam
Project VCM Session Difference Tools Window Help
1
<Show All Types>
<Show Items With Difference...
2
* Starteam8:Documentation-Documentation: Promote from DocDeliverables_ST2009
3
File "ST_MPX_Help.pdf"
File "ST_Server_Help_en.pdf"
File "ST_Toolbar_Help_en.pdf"
Readmes
File "ST_Readme_en.html"
fr
4
OnlineHelp
common
html
File "ST_Install_en.pdf"
File "ST_MPX_Help.pdf"
Folder "Readmes" - Moved in source - Move
File "ST_WhatsNew_en.pdf"
Readmes
File "ST_Readme_en.html"
fr
OnlinebHelp
common
html
images
5
6

Description	Type	Parent	Version	Action
ST_Datamart_Help.pdf	File	Deliverables\Datamart\en\P...	1.2.1.7 --> 1.3	Merge
ST_Datamart_Readme.html	File	Deliverables\Datamart\en\R...	1.1.1.8 --> 1.1	Merge
ST_CA_LDAP_Help.pdf	File	Deliverables\LDAP\en\PDFs\	1.1	Reverse Share
ST_RM_LDAP_Help.pdf	File	Deliverables\LDAP\en\PDFs\ ...	1.3.1.2 --> 1.4	[Deleted in source, modified ...
ST_CA_LDAP_Install.html	File	Deliverables\LDAP\en\Read...	1.2	Reverse Share
ST_CA_LDAP_Readme.html	File	Deliverables\LDAP\en\Read...	1.2	Reverse Share
ST_RM_LDAP_Install.html	File	Deliverables\LDAP\en\Read...	1.2.1.2 --> 1.2	[Deleted in source, target on...
ST_RM_LDAP_Readme.html	File	Deliverables\LDAP\en\Read...	1.2.1.3 --> 1.2	[Deleted in source, target on...
44940	Change Request	Deliverables\Main\ --> Docu...	1.3 --> 1.5	[Modified in target]
45391	Change Request	Deliverables\Main\ --> Docu...	1.6 --> 1.7	[Modified in target]

View - Current | 1,199 Item(s),0 selected | User: Terry Fravel | Manual | Server Release: 11.0.0.37 | EN

1 VCM Menu

2 VCM Toolbar

3 VCM Title Bar

4 Upper Comparison Pane

5 Lower Detail Pane

6 Status Bar

Use the **Compare Perspective** to review what is being compared and what merge actions are planned. The **Compare Perspective** displays the contents of the source and target views in two folder trees at the top.

Selecting an item in a folder tree also selects it in the lower pane. Each comparison pane contains scroll bars and navigation buttons.

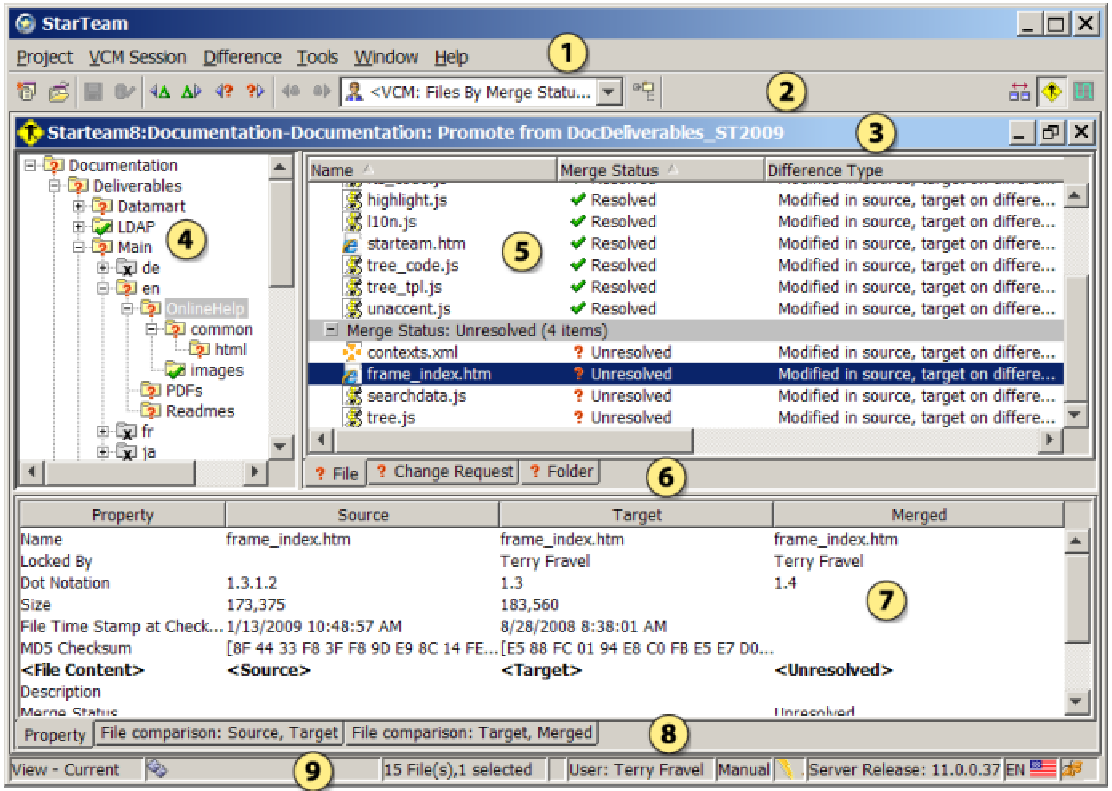
The lower pane in the **Compare Perspective** is a details pane displaying all the items in the two folders. Columns provide the name of the item, item type, parent path, and the version of each item.

The **Version** column shows both the source and target versions of the item if there is a difference. The **Action** column lists the action to be performed for each item. If the item is not <unchanged>, a context menu for the **Action** column enables you to specify a different merge action. The actions are displayed with different formatting indicating the following:

- An action not in brackets indicates the proposed action that View Compare/Merge will take on the item.
- An action in square brackets indicates the compare phase is ignoring the item, taking no action and displays why in the brackets.
- An action in angle brackets indicates no action is set for the item because it was not changed or was not compared.

File comparison and folder comparison in View Compare/Merge highlight differences in color, with each color representing a certain difference type. For example, it highlights inserted or deleted items in green, changed items in yellow, and moved or renamed items in blue. You can modify the default color for differences in the **StarTeamFile Compare/Merge Options** dialog box, accessed from the **StarTeamFile Compare/Merge** window under **Tools** > **Options** .

Merge Perspective



Name	Merge Status	Difference Type
highlight.js	✓ Resolved	Modified in source, target on differe...
l10n.js	✓ Resolved	Modified in source, target on differe...
starteam.htm	✓ Resolved	Modified in source, target on differe...
tree_code.js	✓ Resolved	Modified in source, target on differe...
tree_tpl.js	✓ Resolved	Modified in source, target on differe...
unaccent.js	✓ Resolved	Modified in source, target on differe...
Merge Status: Unresolved (4 items)		
contexts.xml	? Unresolved	Modified in source, target on differe...
frame_index.htm	? Unresolved	Modified in source, target on differe...
searchdata.js	? Unresolved	Modified in source, target on differe...
tree.js	? Unresolved	Modified in source, target on differe...

Property	Source	Target	Merged
Name	frame_index.htm	frame_index.htm	frame_index.htm
Locked By		Terry Fravel	Terry Fravel
Dot Notation	1.3.1.2	1.3	1.4
Size	173,375	183,560	
File Time Stamp at Check...	1/13/2009 10:48:57 AM	8/28/2008 8:38:01 AM	
MD5 Checksum	[8F 44 33 F8 3F F8 9D E9 8C 14 FE...	[E5 88 FC 01 94 E8 C0 FB E5 E7 D0...	
<File Content>	**<Source>**	**<Target>**	**<Unresolved>**
Description			
Merge Status			Unresolved

1 VCM Menu

2 VCM Toolbar

3 VCM Title Bar

4 Folder Tree

5 Upper Pane

6 Component Tabs

7 Lower Pane

8 Information Tabs

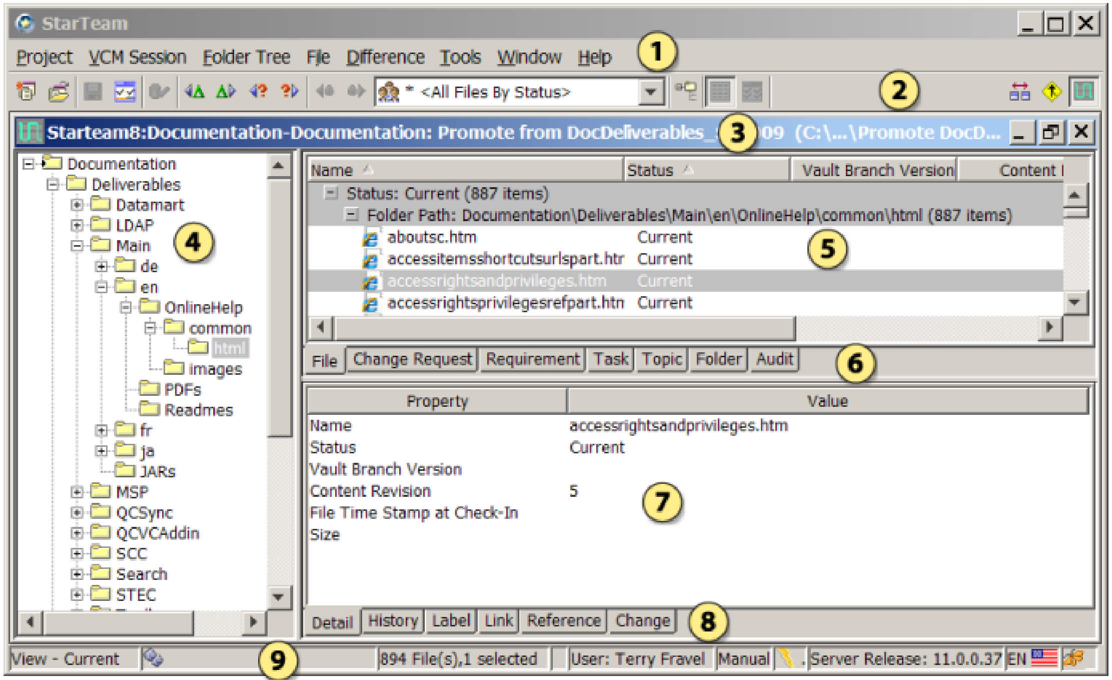
9 Status Bar

Use the **Merge Perspective** to review and adjust merge results. The **Merge Perspective** displays merge status icons in the folder tree showing which folders have items that need to be merged or resolved. The tabs in the upper pane are filtered by **Merge Status** or **Merge Action**, and the **Merge Status** column displays the merge status for each item, such as `Unresolved` and `Resolved`. The icons the upper pane display a flyout menu when you hover over them with the mouse.

The lower pane contains a **Property** tab for displaying the properties of the selected item. For unresolved items, the differences are displayed in boldface type. You can double-click an item or use the context menu on a column to resolve the differences by choosing to use the source version, the target version, or merge the differences. If you choose to merge file content differences, the File Compare/Merge window opens in a three-way merge comparison mode allowing you to automatically, or manually perform the merge. You can also merge text properties.

The lower pane also contains two file comparison tabs, one for comparing a source file with a target file, and one for comparing a target file with the proposed merged results.

Test Perspective



Name	Status	Vault Branch Version	Content I
Status: Current (887 items)			
Folder Path: Documentation\Deliverables\Main\en\OnlineHelp\common\html (887 items)			
aboutsc.htm	Current		
accessitemsshortcutsurlspart.htr	Current		
accessrightsandprivileges.htm	Current		
accessrightsprivilegesrefpart.htn	Current		

Property	Value
Name	accessrightsandprivileges.htm
Status	Current
Vault Branch Version	
Content Revision	5
File Time Stamp at Check-In	
Size	

View - Current | 894 File(s),1 selected | User: Terry Fravel | Manual | Server Release: 11.0.0.37 | EN

1 VCM Menu
2 VCM Toolbar
3 VCM Title Bar
4 Folder Tree
5 Upper Pane
6 Component Tabs
7 Lower Pane
8 Information Tabs
9 Status Bar

The test perspective shows you what the target view would look like if the View Compare/Merge session were committed now. You can to do many of the same StarTeam operations in the **Test Perspective** as you would do in a normal project view, such as checking out files, viewing properties, viewing histories, and comparing changes between revisions.

In the testing phase of a View Compare/Merge session, you can check out files after the compare phase, after auto- and manual-merging has occurred, but before a commit occurs. By default, files checked out from the **Test Perspective** are saved in a temporary View Compare/Merge session working folder in the `[user_home]\Micro Focus\StarTeam\vcmRootDir` directory. You can specify a different working folder if you do not want to use the default. You can run your build tools on this folder to test your changes before committing and closing the session.

Once you have determined that the changes from your View Compare/Merge session are correct, you can commit your changes. View Compare/Merge applies all the changes to the StarTeam repository on the server.

Specifying the Initial Perspective for a View Compare/ Merge Session

1. Open the view to use in your session.
2. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
3. On the **Set Options** page of the wizard, check **Start With Merge Perspective Current** to begin in the **Merge Perspective** (default), or uncheck it to begin in the **Compare Perspective**.

View Compare/Merge Session Filters

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of available filters depends on the perspective you are using in the View Compare/Merge session. StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

You can create additional filters for the **Merge Perspective** and the **Test Perspective**. To access the **Filters** dialog box, right-click a column header in the upper pane of the **Merge Perspective** or **Test Perspective** and choose **Filters**. The **Compare Perspective** has two predefined filter lists, but you cannot create any additional filters for that perspective.

<Show Type>	Displays only items of the selected type in the **Compare Perspective**.
<Show Items With Differences>	Displays items from both the source and target views in the **Compare Perspective** that have differences.
<Show Unresolved Items>	Displays only items in the **Compare Perspective** with a merge status of **Unresolved**.
<Show All Items>	Displays all items of the selected type in the **Compare Perspective**.
Show Items: [action]	Displays only items in the **Compare Perspective** with the selected merge action.
<VCM: Items By Merge Status>	Displays items in the upper pane of the **Merge Perspective** sorted by merge status. This filter name is dynamic, depending on which component tab is selected. You can display items by merge status for files, change requests, and folders.
<VCM: Merge Properties>	Displays the items in the upper pane of the **Merge Perspective** with the primary descriptor in column one, and all the remaining VCM properties in the following columns.
Merge Action	Sorts items in the upper pane of the **Merge Perspective** and **Test Perspective** the by merge action.
<Show Items With Updates>	Displays only items from the source view in the **Compare Perspective** that will result in updates to the target.

Note: In the **Test Perspective**, you also see the filters for files when the **Files** tab is selected, for folders when the **Folders** tab is selected, and so on.

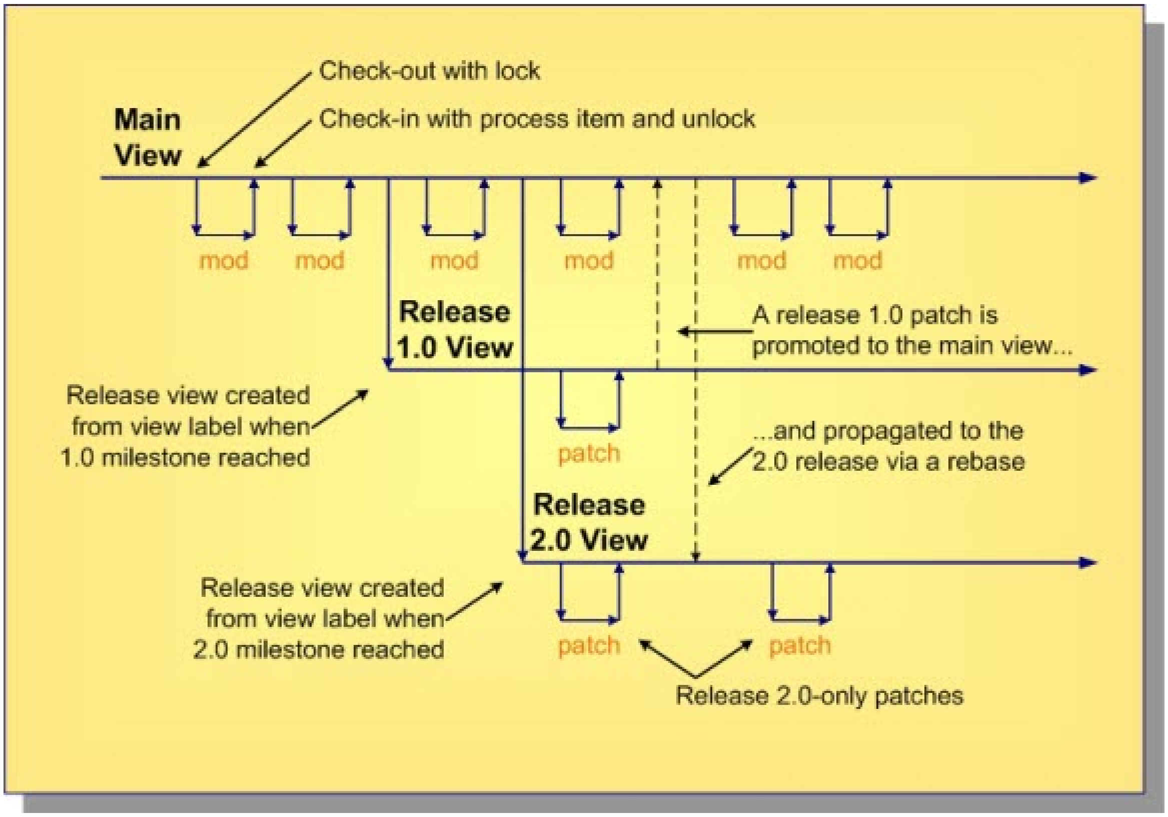
Lifecycle Scenarios in View Compare/Merge

The topics below present examples of some typical full-lifecycle scenarios using View Compare/Merge for cross-view change management.

View Compare/Merge Usage with a Simple Project

In this example, you have a small project and a small team, making it is generally safe (for example, not disruptive) to perform new development directly in the main view. The only other views you may need are release views for maintenance purposes. In this scenario, your primary use of View Compare/Merge is to propagate patches from or to the release views.

Below is an example of using View Compare/Merge with a simple project.


Check-out with lock
Check-in with process item and unlock
Main View
mod
mod
mod
mod
mod
mod
Release 1.0 View
A release 1.0 patch is promoted to the main view...
Release view created from view label when 1.0 milestone reached
patch
...and propagated to the 2.0 release via a rebase
Release 2.0 View
Release view created from view label when 2.0 milestone reached
patch
patch
Release 2.0-only patches

In this scenario, you perform all new development with direct modifications in the main view. When a release milestone is reached, you create a view label as the reference point for defining a release view. To fix bugs in a given release, you make changes directly in the release view. If you make a patch in one release that must be propagated to the main view and another release, first deliver it to the main view using a View Compare/Merge promote, then propagate it to the second release with a View Compare/Merge rebase. (Alternatively, if the patch did not apply to the main view, you could use a View Compare/Merge replicate to propagate it directly from one release view to another.)

Using View Compare/Merge with Activity and Sandbox Views

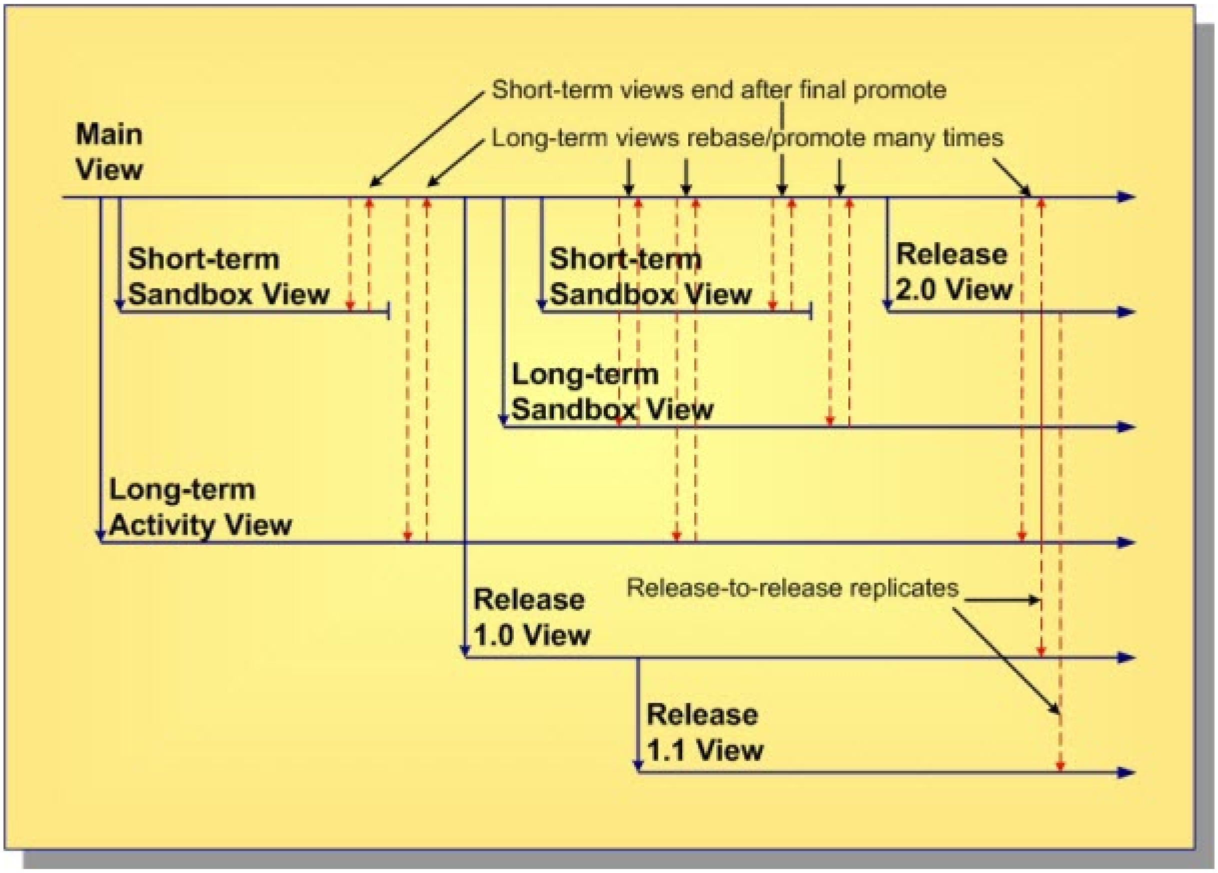
In most circumstances, direct modification to the main view is not advisable due to the potential instability it may cause. Activity and sandbox views provide a place where development can occur without disrupting the stability of the parent view. These view roles also provide a way to discard work when there is a chance that it could be cancelled and never delivered to the main view.

The primary difference between activity and sandbox views is scope:

- Activity views generally serve a group of developers and a specific purpose such a single, large enhancement.
- Sandbox views generally serve an individual developer or a very small team and can be used for many iterations of bug fixes and enhancements.

Both activity and sandbox views can have short or long life spans.

When activity and sandbox views are used, all new, fully-verified and approved changes should make their way to the main view. Release views should be created from the main view to provide streams for maintenance work. It even makes sense to sometimes create a release view from another release view, for example when a “dot release” will be shipped then maintained separately from its parent release. An example of recommended View Compare/Merge best practices with activity and sandbox views is shown below.



In this example, two short-term sandbox views are created from the main view. (The “mod” and “patch” cycles shown in the previous example are omitted here to reduce clutter.) After the work is complete in these views, they are promoted to the main view and become inactive (and eventually deleted). A long-term activity view and a long-term sandbox view are also shown. These views “live” beyond a single release. Therefore their work may be promoted to the main view several times.

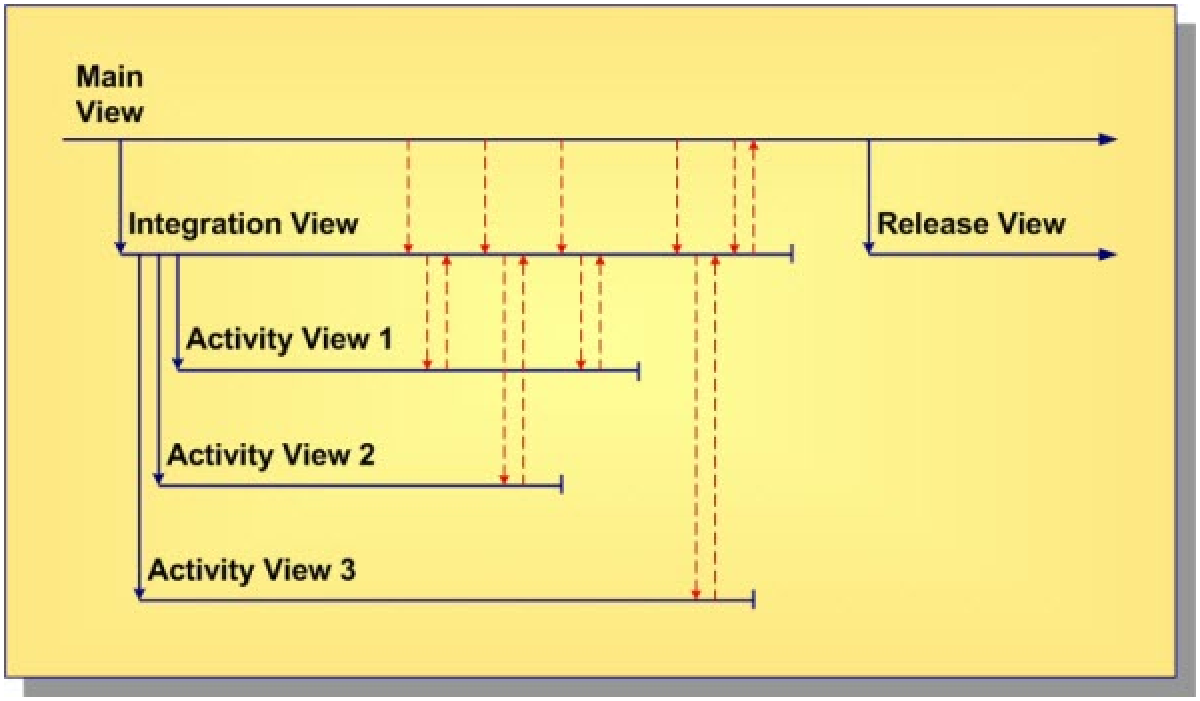
This example demonstrates the recommended best practice of always rebasing a view just prior to promoting it. This causes the child view to merge changes from the parent so they can be integrated and tested.

This example also shows release `1.0` and `2.0` views being created from the main view, but a release `1.1` view is created as a child of the release `1.0` view. Also shown is a patch made in the release `2.0` view and being “back ported” directly to the release `1.0` and `1.1` views using replicate merges.

Using View Compare/Merge in High Concurrency Projects

Suppose you have a large project involving many sub-teams, each developing a separate component of a final application or suite. You may find it impractical to have each team use its own activity view and then integrate the changes in the main view. The integration and stabilization of large projects can be lengthy, causing long periods of instability in the main view, which is not good.

To address the needs of large development projects, we introduce the notion of an *integration view*. An integration view is a branch-all view that is positioned between the main view and one or more activity views. When the time is right, the “grandchild” activity views are promoted to the integration view, where integration testing occurs. When the integration view passes all applicable tests, it is finally promoted to the main view, delivering its changes. An example using integration views is shown below.



Notice that the integration view is created first as a child of the main view. An activity view for each major component or sub-team is then created as a child of the integration view. The reason for creating the integration view first as a parent of the activity views is so that View Compare/Merge promote merges can be used to propagate changes from the activity views. Promote is the best merge technique for new development because it uses rules that are designed for propagating changes "up" the view hierarchy. For example, new items are reverse-shared instead of shared, causing the parent view to take ownership of the main (`1.x`) artifact branches. If the integration view was created after the activity views, it could not be their parent, so you would have to propagate changes with VView Compare/MergeCM *replicate*, which uses *share* as the default action for new items instead of *reverse-share*.

If changes occur in the main view while work is done the activity views, the activity views should periodically be rebased to accept those changes. However, they cannot be rebased from the main view since it is not their parent. Instead, you must rebase the integration view first. Rebasing the integration view should be very easy (no merge conflicts should be found) since no changes should occur directly in the integration view until the integration work is ready to begin.

After an activity view has been rebased, it can be promoted to deliver its changes to the integration view. You could immediately retire an activity view once it has been promoted, choosing to resolve any integration issues and fix bugs in the integration view. However, for complex integrations, you may choose to promote, rebase and adjust, then promote an activity view several times. Eventually, though, you should shift the integration effort to the integration view since it has the big picture. When the integration view passes all tests, promote it the to main view, and create a release view.

Merge Types, Rules, and Scenarios

This topic presents an overview of the different View Compare/Merge merge types you can use, explains the rules that apply to them, and gives you example usage scenarios for each one.

Merge Types

There are three types of merge operations possible for a View Compare/Merge session:

- Rebase
- Promote
- Replicate

Each merge type has specific rules that apply to it.

You can perform successive Rebase, Promote, and Replicate operations between the same two views, even with reversing the direction for Replicate. You can only use one merge type for each View Compare/Merge session, but you can perform as many View Compare/Merge sessions as you like in any order.

View Compare-only Mode

View Compare/Merge provides a Compare-only mode in which you can view the differences between views, but no merge actions or changes are possible.

In Compare-only mode, View Compare/Merge treats the source view, the target view and the View Compare/Merge session itself as read-only. Since no changes are made to the source or the target, Compare-only mode supports comparisons that are not otherwise permitted. Specifically, you can compare two different configurations of the same view. For example, you might compare two different labeled configurations, or compare a labeled configuration to the tip.

When you only want to view the differences between two views, you can run a Compare-only View Compare/Merge session. In this case, View Compare/Merge does not determine any default or possible actions to take, since no actions are requested. The Compare-only View Compare/Merge session simply shows you the differences between the items in the source and target views.

When to Use Each Merge Type

Each of the View Compare/Merge merge types is intended to be used in specific scenarios. Which merge type you choose for a particular View Compare/Merge session depends on what you are trying to merge. Using View Compare/Merge for the right scenarios is important because View Compare/Merge uses different rules for handling differences depending on the type of merge. Below is a description of the intended use for each View Compare/Merge merge type and the rules that apply.

Rebase

Use a Rebase merge to “catch-up” a child view with changes that have occurred in the parent view since the child view was created, or since the last Rebase.

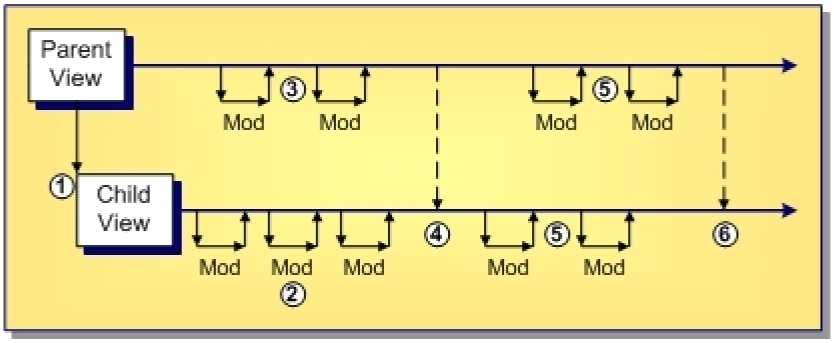
Rules for Rebase

When performing a Rebase merge operation, follow these general rules:

- The target view must be a branching (variant), immediate child of the source view.
- The target view items should have branch on change set, and have been created using a pinned (not floating) configuration.
- The Rebase operation can use any configuration of the source view (tip, label, timestamp, or promotion state), called the *Rebase point*.
- The Rebase operation uses the tip (updatable) configuration of the target view.
- Items newly-shared to or re-pinned in the target view use the Rebase/Replicate point. That point is the configuration time of the item in the source view.

Rebase Example Scenario

This example shows a typical scenario where an activity (child, branch-all variant) view is created from a specific label, promotion state, or timestamp to support a specific bug fix or enhancement. After changes are made in parallel in the parent and the child, a Rebase is performed to propagate parent changes to the child view. This process is repeated until the child view is no longer needed.



1. An activity (child, branch-all variant) view is created from a specific label, promotion state, or timestamp to support a specific bug fix or enhancement.
2. Changes are made for the activity: modified files branch, new files are added, files are moved or renamed, and so on.
3. Changes are made in parallel in the parent view.
4. A Rebase is performed to propagate changes in the parent to the child view.
5. More changes occur in both views in parallel.
6. Another Rebase is performed to propagate new changes.

In this scenario, an activity view is rebased twice to update it with changes that occurred in the parent.

Promote

Use a promote merge to “deliver” changes in a child view to its parent. You could promote once at the end of the child view’s lifecycle as a prelude to discontinuing work in the child view. However, you can also promote multiple times to periodically deliver changes from the child view to the parent. In either case, it is typically a good idea to rebase the child view just prior to promoting it to ensure it has “accepted” (been merged with) all recent changes from the parent view.

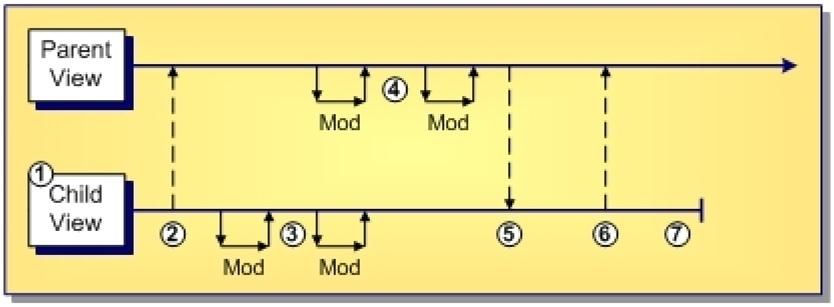
Rules for Promote

When performing a Promote merge type, follow these general rules:

- The source view must be a branching (variant), immediate child of the target view.
- The Promote operation uses the tip (updatable) configuration of the target view.
- The Promote operation can use any configuration of the source view (tip, label, timestamp, or promotion state).
- New source items are *reverse-shared* to the target view. This means they are moved from the child to the parent and then shared back to the child with **Branch-On-Change** set, and with a pinned configuration. New items not promoted at the latest (tip) revision may be marked as merge conflicts in the child view.

Promote Example Scenario

This example shows a typical scenario where the a child view is created to enable a new branch of development that is simultaneous with continued development on the main branch. During development, the child view is modified, rebased, compiled, tested, then ready for release. At various stages, changes from the child view are promoted to the parent view to release changes, perhaps at key delivery milestones. At the end of the branch's life, a Rebase is performed with the parent view. After the changes in the child from the parent view are compiled, the child view is promoted to the parent again.



1. An "activity" view has been created, modified, rebased, compiled, tested, and is ready to be released.
2. A Promote is performed to release child changes to the parent view.
3. More changes are needed in the activity view.
4. Meanwhile, the parent is also modified.
5. A Rebase catches up the child with the parent.
6. Compile and test are okay, so the child is promoted to the parent again.

7. The activity view eventually becomes obsolete after that.

In this scenario, the child view is promoted twice and retired (no longer used) after the second promote. Typically, you'd want to delete a retired view after a period of time.

Replicate

Replicate duplicates changes from one view to another. Use a replicate merge to propagate a change when the target view is not an immediate parent or child of the source view.

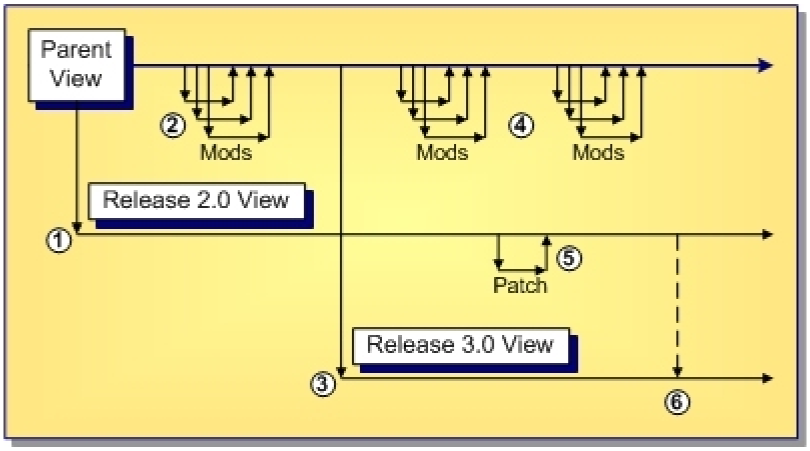
Rules for Replicate

When performing a Replicate merge type, follow these general rules:

- The source and target views need to belong to the same project. They should have some common lineage, such as parent/child, siblings, uncle/niece, and so on.
- The Replicate operation can use any configuration of the source view (tip, label, timestamp, or promotion state), called the *Replicate point*.
- The target view must be an updatable, branching (variant) view, and the tip configuration must be used.
- Items newly-shared to or re-pinned to in the target view use the Rebase/Replicate point.

Replicate Example Scenario

This example shows a typical scenario where the parent view was branched twice for different product releases, creating `2.0` and `3.0` branches. During product maintenance, changes are made in the `2.0` branch that are applicable to the `3.0` branch, but not to the main branch. A Replicate is performed to duplicate the changes from `2.0` to `3.0`.



1. A "release" (child, branch-all variant) view is created to support the `2.0` release.
2. Changes are made in the main view in preparation for the next release.
3. A "release" view is created to support the `3.0` release.
4. More changes are made for future releases.
5. A bug is fixed in `2.0` that is applicable to `3.0`, but not other releases.
6. A Replicate is performed to duplicate the patch from the `2.0` view to the `3.0` view.

In this example, a bug fix must be propagated from one release view to another, but the release views are old with respect to the tip revision of the parent view and not applicable to it. Consequently, it makes sense to merge the fix “sideways”, directly from the source view to a sibling target view.

Tips for Successful Sessions

Prepare Data

StarTeam allows you to do almost anything you want to your data. However, this can result in conditions which a View Compare/Merge session cannot make decisions without your input. It is best to get your data into the best possible shape and keep it there.

- When a branching item (folder, file, or change request) is the root branch, or is the first share of a new branch, (`1.x`), its behavior should be disabled, and its configuration should be floating. Pinning its configuration makes it read-only.
- When not the first share of its branch, branching items (folders, files, and change requests) should be set to branch on change (behavior), and have a pinned configuration.

Some customers make exceptions for this. For example, they might want all change requests visible from all views, regardless of where they were created. So, they share the same change request folder into each child view, but they set the folder and the change requests to not branch on change and to be floating. This is not a best practice, but can be used, carefully. Because the change request never branches, when they use a change request from any view as a process item, they are essentially using an out-of-view process item from the main view. That means that they should be using enhanced process links (like all users who use out-of-view process items).

Other customers put all change requests in the main view and always select them as out-of-view process items. They usually make a copy of each change request for each maintenance or release view in which the change request needs to be fixed. This allows each change request for a given problem to take separate workflow steps, or, the same steps but at different times. The copies are also in the main view.

Follow Best Practices

If you follow best practices, the only time you will run into problems is when you share items into a new location. This gives the following results:

- The shared items have the behavior determined by the view property **Set items shared into view to branch on change** check box, which we recommend to be checked. This property is ignored for items that cannot branch.
- When dealing with an item that cannot branch (a task, requirement, or topic), pinning its configuration makes it read-only, so you want to use it in only one view at a time. For example, for a task that is created in an activity view, it often makes sense to promote the task along with the changed files. Once the task is promoted, it becomes read-only in the activity view, and you will use it in the parent. This continues up the view hierarchy until the task and its associated changes have been promoted to the root.
- The shared items will be floating. You will need to set their behaviors to branch on change.

Note: For folders, you select the root folder of the newly shared items (if the share contains folders) and set the configuration for the folder and its subtree of folders. For the item types in the share, you must select them all from the upper pane (type-by-type), and change their configuration from floating to pinned. Usually customers pin this to the current time, and usually they are primarily concerned with files.

View Compare/Merge Actions

When you start a View Compare/Merge session, it automatically compares the contents of the source and target views using the options you selected in the **View Compare/Merge Wizard**. Wherever possible, View Compare/Merge suggests a default action to apply to the target view for each item with differences. However, View Compare/Merge is unable to automatically resolve or merge some types of differences, such as certain conflicts between file contents. If it cannot determine an action, or merge the differences, it gives the items an **Unresolved** status and proposes a **Merge** action. For **Merge** items, you must perform a manual merge or change the action to one that View Compare/Merge can perform automatically, such as **Ignore**.

You can override the default action proposed by VCM for both **Resolved** and **Unresolved** differences. When VCM cannot suggest a resolution, the action is **Fail** and must be overridden.

View Compare/Merge can perform many different actions during a session, but it only allows specific actions appropriate for each type of change. For example, if an item is new in the source and does not exist in the target, you can ignore the item, but you cannot delete it from the target. **Delete**, in this case, is not a possible action.

The following table describes each of the actions View Compare/Merge can perform for a given source/target item difference.

Note: When you create a custom View Compare/Merge merge type with the **Customize VCM** tool in the **Server Administration** tool, a `custom.vcm.xml` file is created and each merge action is represented by an enumerated value which is shown in the following table.

Enumerated Value	Action	Description
0	`Ignore`	Take no action.
1	`Re-pin`	Changes the revision to which the target is pinned to match the source item.
2	`Move`	Moves the target item to the equivalent folder as the source item.
4	`Merge`	Merges the source and target items. The target item is modified to reflect the merged result.
8	`Delete`	Deletes the target item.
16	`Share`	Shares the source item to the target view.
32	`Reverse Share`	Moves the source item to the target view and share it back to the source view.
64	`Overwrite`	Overwrites the target with the contents of the source.
66	`MoveAndOverwrite`	Equivalent to a `Move` followed by an `Overwrite`.
256	`Mark Resolved`	Creates a `Mark Resolved` Change Object. *Mark Resolved* is now a legal action in many cases where it was not previously permitted. This affects about two dozen difference types.
6	`Move and Merge`	Equivalent to a `Move` followed by a `Merge`.
3	`Move and Re-pin`	Equivalent to a `Move` followed by a `Re-pin`.
40	`Delete and Reverse Share`	Equivalent to a `Delete` followed by a `Reverse Share`.
128	`Needs Review`	`Needs Review` is a legal action in all cases. View Compare/Merge cannot determine what action to take and will not commit changes while items are set to `Needs Review`.You must set `Needs Review` items to another action, one that can be performed. Changing the default action of any difference type to *Needs Review* means that human intervention is required before a commit can be made.

Not every action is valid for every item difference. For example, `Delete` is not valid when the target item is already deleted.

Note: No **Merge** or **Mark As Resolved** actions are possible for requirements, tasks, and topics, which are times which cannot be branched.

Conditions for Action Decisions

This topic explains the actions View Compare/Merge can take to determine which set of conditions apply to the source and target item differences during a session.

View Compare/Merge looks at the state of each item being compared during the compare phase of a session and determines which set of conditions apply to each item. It matches the state of the items being compared to a set of predefined conditions, and chooses an action to take for resolving the conflict.

In more complicated situations, it is unable to determine an action to take. In this case, you must specify the action View Compare/Merge should take to resolve the differences. You may also override the default action chosen to resolve a conflict.

Below are the conditions for which View Compare/Merge tests:

Conditions Tested	Description
Whether the item already exists in either the source or target views.	These conditions are represented by the **Present** columns of the Decision Table
Whether the item has been deleted from either the source or target views.	These conditions are represented by the **Deleted** columns of the Decision Table.
Whether the item has been modified in either the source or target views.	These conditions are represented by the **Changed** columns of the Decision Table.
Whether the item has been moved in either the source or target views.	These conditions are represented by the **Moved** columns of the Decision Table.
Whether the source or target item is the root branch of its share tree	These conditions are represented by the **Root Share** columns of the Decision Table
Whether the source item is a child share of the target item Whether the target item is a child share of the source item.	These conditions are represented by the **Child Share of Twin** columns in the Decision Table and refer to an immediate child, not a grandchild, of the matching--that is twin--item in the other view.
Whether the source or target items have a floating configuration	These conditions are represented by the **Floating** columns of the Decision Table.
Whether the target folder has floating shares in the source view.	This condition is represented by the **Target folder has floating shares** column of the Decision Table.
Whether the target item's folder has been deleted.	This condition is represented by the **Target Parent Deleted** column of the Decision Table.
Whether the target item's folder has been ignored by the session.	This condition is represented by the **Parent Folder Ignored** column of the Decision Table.
Whether the target item's folder has failed in the session.	This condition is represented by the **Parent Folder Failed** column of the Decision Table.
Whether the file (if the item in question is a file) is binary.	This condition is represented by the **Items are Binary** column of the Decision Table.
Whether the source and target items have the same user-modifiable properties and, for files, the same data content.	This condition is represented by the **Identical Contents** column of the Decision Table.
Whether the properties that differ between the two items have been excluded.	This condition is represented by the **Excluded Properties** column of the Decision Table.

View Compare/Merge Table of Action Decisions

When comparing views in a VCM session, VCM compares the state of each pair of items in the source and target views against a table of difference types to determine if the items match the criteria in any of the rows. From that, it decides what, if any, are the default and possible actions for resolving each item difference for a given merge type. The resulting list of default and possible actions is what is presented during the VCM session.

The following table shows the rules VCM uses to decide which actions to take under a given set of conditions for each merge type: `Promote`, `Rebase`, and `Replicate`. The table shows all the possible actions that can be taken in a VCM Session. In the table, the default action is always listed as the first action. Ignoring the item differences is always one of the options.

This table explains how VCM will treat each pair of matching items, one in the source and one in the target. Because the table indicates what actions VCM allows and what action VCM will select by default, it can be considered a set of use cases. The actions allowed by VCM depend on the type of operation (Rebase, Promote, or Replicate), and the following characteristics of the items:

- Whether an item in the source matches an item in the target (that is, whether the item exists in only one or both views).
- Whether the matching items are on the same branch or different branches (based on their dot notations).
- Whether the corresponding items are in the same location in the two views (that is, in the same StarTeam folder path. A difference in location indicates that one or both of the matching items has been moved).
- Whether additional revisions of the matching items have occurred since the last merge.
- Whether the target item is a floating child of the source item.

Note: The common ancestor of the matching items or the merge point set for the matching items is used to determine if changes have been made to the source and target items.

ID	Description	Replicate	Rebase	Promote	Possible Actions
100	Parent Folder Failed.	x	x	x	Needs Review, Ignore.
110	Parent Folder Ignored.	x	x	x	Ignore, Needs Review.
200	Target folder has floating Share in source view.	x	x	x	Needs Review, Ignore (overlayed on other rows).
620	Deleted in target (Promote).			x	Ignore, Needs Review, Reverse Share, Share.
600	Deleted in target.	x	x		Ignore, Needs Review, Share.
520	New in source, shared (Promote).			x	Needs Review, Ignore, Share.
510	New in source (Promote).			x	Reverse Share, Ignore, Needs Review, Share.
500	New in source.	x	x		Share, Ignore, Needs Review.
800	Deleted in source.	x	x	x	Delete, Ignore, Needs Review.
900	Deleted in source, target on different branch.	x	x	x	Ignore, Delete, Needs Review.

ID	Description	Replicate	Rebase	Promote	Possible Actions
1000	Deleted in source, modified in target.	x	x	x	Ignore, Delete, Needs Review.
700	New in Target	x	x	x	Ignore, Delete, Needs Review.
1230	Floating child root share, source unbranched (Promote).			x	Delete-and-Reverse-Share, Ignore, Mark Resolved, Needs Review, Repin.
1220	Floating child share (Promote).			x	Needs Review, Mark Resolved, Ignore, Repin.
1210	Floating child share.	x	x		Repin, Ignore, Mark Resolved, Needs Review.
1200	Identical in source and target.	x	x	x	Ignore, Needs Review.
1300	Unchanged since last merge.	x	x	x	Ignore, Needs Review, Overwrite.
1410	Modified in source, target is child share of source (Promote).			x	Delete-and-Reverse-Share, Ignore, Mark Resolved, Needs Review, Repin.
1405	Modified in source, same content.	x	x	x	Mark Resolved, Ignore, Needs Review, Repin.
1408	Modified in source, some changed properties excluded.	x	x	x	Merge, Ignore, Mark Resolved, Needs Review, Overwrite, Repin.
1400	Modified in source.	x	x	x	Repin, Ignore, Mark Resolved, Needs Review.
1510	Modified in source, target on different branch, same content.	x	x	x	Mark Resolved, Ignore, Needs Review.
1500	Modified in source, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Needs Review, Overwrite.
1610	Modified in source, modified in target, same content.	x	x	x	Mark Resolved, Ignore, Needs Review.
1600	Modified in source, modified in target.	x	x	x	Merge, Ignore, Mark Resolved, Needs Review, Overwrite.
1720	Modified in target, same content.	x	x	x	Ignore, Mark Resolved, Needs Review.
1700	Modified in target.	x	x	x	Ignore, Mark Resolved, Needs Review, Overwrite.
1900	Moved in source.	x	x	x	Move, Ignore, Mark Resolved, Needs Review.
2000	Moved in source, target on different branch.	x	x	x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Overwrite, Needs Review.

ID	Description	Replicate	Rebase	Promote	Possible Actions
2050	Moved in source, modified in target.	x	x	x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Overwrite, Needs Review, Overwrite.
2060	Moved in source, branched and modified in target.	x	x	x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Needs Review, Overwrite.
2100	Moved in source, moved in target.	x	x	x	Ignore, Mark Resolved, Move, Needs Review.
2150	Moved in source, moved and modified in target.	x	x	x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Needs Review, Overwrite.
2200	Moved in target.	x	x	x	Ignore, Mark Resolved, Move, Needs Review.
2205	Moved and modified in target.	x	x	x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Needs Review, Overwrite.
2410	Moved and modified in source (Promote).			x	Ignore, Mark Resolved, Move, Move-and-Overwrite, Needs Review, Overwrite.
2420	Moved and modified in source, same content.	x	x		Move, Mark Resolved, Ignore, Move-and-Repin, Needs Review, Repin.
2408	Moved and modified in source, some changed properties excluded.	x	x	x	Move-and-Merge, Ignore, Mark Resolved, Merge, Move, Move-and-Overwrite, Move and Repin, Needs Review, Overwrite, Repin.
2400	Moved and modified in source.	x	x		Move-and-Repin, Ignore, Mark Resolved, Move, Needs Review, Repin.
2530	Moved and modified in source, target on different branch, modified in target, same content.	x	x	x	Mark Resolved, Ignore, Move, Needs Review.
2520	Moved and modified in source, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Merge, Move, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2510	Moved and modified in source, target on different branch, same content.	x	x	x	Mark Resolved, Ignore, Move, Needs Review.
2500	Moved and modified in source, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Move, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2630	Moved and modified in source, moved in target, target on same branch, same content.	x	x	x	Move, Ignore, Mark Resolved, Move-and-Repin, Needs Review, Repin.
2610	Moved and modified in source, moved in target,	x	x	x	Mark Resolved, Ignore, Move, Needs Review.

ID	Description	Replicate	Rebase	Promote	Possible Actions
	target on different branch, same content.				
2600	Moved and modified in source, moved in target, target on different branch.	x	x	x	Merge, Igonre, Mark Resolved, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2615	Moved and modified in source, moved and modified in target, target on different branch, same content.	x	x	x	Mark Resolved, Ignore, Move, Needs Review.
2805	Moved and modified in source, moved and modified in target, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Move, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2910	Moved and modified in source, moved in target, target is child share of source (Promote).			x	Delete-and-Reverse-Share, Ignore, Mark Resolved, Move, Move-and-Repin, Needs Review, Repin.
2908	Moved and modified in source, moved in target, some changed properties excluded.	x	x	x	Move-and-Merge, Ignore, Mark Resolved, Merge, Move, Move-and-Overwrite, Move and Repin, Needs Review, Overwrite, Repin.
2900	Moved and modified in source, moved in target.	x	x	x	Move-and-Repin, Ignore, Mark Resolved, Move, Needs Review, Repin.
2730	Modified in source, moved in target, same content.	x	x	x	Mark Resolved, Ignore, Move, Move-and-Repin, Needs Review, Repin.
2710	Modified in source, moved in target, target on different branch, same content.	x	x	x	Mark Resolved, Ignore, Move, Needs Review.
2700	Modified in source, moved in target, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Move, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2715	Modified in source, moved and modified, target on different branch, same content.	x	x	x	Mark Resolved, Ignore, Move, Needs Review.
2705	Modified in source, moved and modifed in target, target on different branch.	x	x	x	Merge, Ignore, Mark Resolved, Move, Move-and-Merge, Move-and-Overwrite, Needs Review, Overwrite.
2810	Modified in source, moved in target, target is child share of source (Promote).			x	Delete-and-Reverse-Share, Ignore, Mark Resolved, Move, Move-and-Repin, Needs Review, Repin.

ID	Description	Replicate	Rebase	Promote	Possible Actions
2808	Modified in source, moved in target, some changed properties excluded.	x	x	x	Merge, Ignore, Mark Resolved, Move, Move-and-Merge, Move-and-Overwrite, Move-and-Repin, Needs Review, Overwrite, Repin.
2800	Modified in source, moved in target.	x	x	x	Repin, Ignore, Mark Resolved, Move, Move-and-Repin, Needs Review.
9900	Unrecognized	x	x	x	Needs Review, Ignore.

Note: `Merge` or `Mark As Resolved` actions are not possible for non-branchable items (requirements, tasks, and topics).

Understanding View Compare/Merge Difference Types

This topic gives a detailed explanation about some of the difference types you will encounter in a View Compare/Merge session.

Below are some of the difference types you may encounter during View Compare/Merge sessions and their corresponding row numbers. This section discusses what to do when the difference type results in the default merge action **Fail**. Unless another merge action is selected for items that **Fail**, the View Compare/Merge session cannot be committed. Often you want to make some data repairs in these cases and then redo the View Compare/Merge session. Sometimes you will simply ignore the situation temporarily so that you can commit other files. Later, you can perform a repair.

Parent Folder Failed (Row 100)

Defaults to **Fail**.

This difference type is applied dynamically. Whenever an item resides in a folder to which a View Compare/Merge session has already assigned the merge action **Fail**, then that item also fails, but with difference type **Parent Folder Failed (Row 100)**. If you change the parent folder's merge action, the merge action for the items within that folder will also change.

For example, suppose an item A has the difference type **Parent Folder Failed (Row 100)** because its parent folder B had difference type **Target folder has floating child share in source view (Row 200)**. If you change B's merge action from the default **Fail** to **Ignore**, the parent folder no longer fails. Item A's difference type also changes to **Parent Folder Ignored (Row 110)**. If you want to commit changes to item A, you must fix folder B so that it is not floating in the target view, refresh the view, and redo the View Compare/Merge session.

Parent Folder Ignored (Row 110)

Defaults to **Ignore** which is also the only available action.

This difference type is applied dynamically. Whenever an item resides in a folder to which a View Compare/Merge session has already assigned the merge action **Ignore**, then that item also ignored, but with difference type **Parent Folder Ignored (Row 110)**. This allows you to commit the other resolved items.

However, you can correct the parent folder and also commit the items with this difference type. For example, suppose an item A has the difference type **Parent Folder Ignored (Row 110)** because its parent folder B had difference type **Deleted in target (Promote) (Row 620)** or **Deleted in target (Row 600)**. If the deleted folder belongs back in the target view, you would change B's merge action from the default **Ignore** to **Share** or **Reverse Share**. Because the parent folder issue has been resolved, item A's difference type changes from **Parent Folder Ignored (Row 110)** to a more appropriate difference type. For example, this item may also be **Deleted in target (Row 600)** or it may be **New in Source (Row 500)** In either of these cases, you may now change item A's merge action to **Share** or **Reverse Share**.

Unlike the **Fail** difference type, **Parent Folder Ignored (Row 110)** usually does not require you to make changes to the repository. Instead, this is used in cases where **Ignore** is the only thing that makes sense, based on a decision you have already made about the parent folder. For example, suppose you have a file that is **New in source (Row 500)**, but the parent folder was **Deleted in target (Row 600)**. The default action for the folder is **Ignore**, which implies that there is no parent folder in the target into which the new file can be shared. That is why the file gets the **Parent Folder Ignored (Row 110)** difference type. If you override the action on the folder to **Share**, the file becomes **New in source - Share**. The **Parent Folder Ignored (Row 110)** difference type no longer applies.

Floating Child Share (Rows 1210, 1220, and 1230)

This section discusses **Floating Share** difference types: **Floating Child Share (Rows 1210)**, **Floating Child Share(Promote) (Rows 1220**, and **Floating Child Share, Source Unbranched (Promote) (Rows 1230)**. These types occur only when you use the **Fix floating child shares** option in the **View Compare/Merge Wizard**.

The intent of the **Fix floating child shares** option in the **View Compare/Merge Wizard** is to locate items in the target view that:

- Are on the same branch.
- Have their parent reference in the source view.
- Are floating.

The operation type affects which difference type is selected by View Compare/Merge. If you are not doing a promote operation, the difference type is **Floating Child Share (Row 1210)**. If you are doing a promote operation, the difference type is **Floating Child Share, Source Unbranched (Promote) (Rows 1230)**. On earlier releases of StarTeam Server, you will see **Floating Child Share (Promote) (Row 1220)** because earlier releases were not able to reverse share items in as many circumstances.

Note: Because of the relationship in the reference tree for the corresponding items in this case, you will never see this difference type when the corresponding items have no common ancestor and were matched only by name.

A floating child item can cause problems. For example, adding an item to a floating child folder in the target can result in that item floating back to the source as a duplicate. Adding a new revision to an item that is floating in the target can result in that revision inadvertently floating back to the source. If the entire view is branch-all floating or branch-none floating, this can happen frequently. That is why StarTeam, depending on how your administrator has set up the StarTeam Server, now either:

- Restricts floating views permanently.
- Makes them advanced features to keep them to a minimum.

Floating Child Share (Row 1210)

Defaults to **Repin**. Optionally, you can select **Ignore**.

The best and most common practice in StarTeam is for *parent* > *child* relationships in the share tree to match the *parent* > *child* relationships in the view hierarchy.

This difference type occurs only when you check the **Fix floating child shares** option in the **View Compare/Merge Wizard**. You use this option to stop the creation of unwanted duplicates in the source as an item floats from the target back into the source.

Tip: It is usually better to correct your data before using View Compare/Merge so that you do not have inappropriate floating shares. Unwanted duplicates can float into more views that the source view.

The intent of the **Fix floating child shares** option in the **View Compare/Merge** wizard is to locate items in the target view that have their parent reference in the source view, and are also floating. Because of this, the rebase operation finds items in the child view that correspond to items in the parent view, and pins them if they are floating. The difference type is **Floating Child Share (Row 1210)**. If you allow View Compare/Merge to re-pin in this case, it changes the floating items to the configuration time of this particular View Compare/Merge session when the session is committed. This is a good idea because you will not have problems with these items in the future.

If you are performing a replicate operation, the references in the reference tree and the views in the view tree may not be pointing in the same direction (from parent to child), but the results will be the same. View Compare/Merge offers to repin these items in spite of their reference relationship because the two views in the View Compare/Merge session are probably not parent and child anyway.

Note: This difference type has limitations: While View Compare/Merge changes the floating child item from a floating to a pinned configuration, it only pins items in the target view. Other views may have floating references for these items. Duplicates can occur in those other views if you proceed with the commit.

For example, suppose a folder was shared from the parent view to each of two child views, set to branch on change in each child view, but allowed to float in the child views. Because View Compare/Merge alerted you, you changed the folder in the first child view so that it is now pinned. Nevertheless, if you promote an item from that folder in the now corrected child view to the parent view, the item will float down to the other child view from the parent view if no intervention has taken place in that view.

If the creation of floating views was common practice, it is best to pin items in all the project's child views before a View Compare/Merge commit operation. It is also a good practice to pin floating items after they are shared to a new location.

Floating Child Share, Source Unbranched (Promote) (Row 1230)

Defaults to **Delete-and-Reverse-Share**. Other merge actions available are **Ignore** and **Repin**.

This difference type occurs only in promote operations when you check the **Fix floating child shares** option in the **View Compare/Merge** wizard. You use this option to stop the creation of unwanted duplicates in the source as an item floats from the target back into the source.

Tip: It is usually better to correct your data before using View Compare/Merge so that you do not have inappropriate floating shares. Unwanted duplicates can float into more views than the source view.

This difference type allows View Compare/Merge to do more than **Fail** when a floating share is found during a promote operation. If you perform a **Delete-and-Reverse Share**, the reference tree is corrected. Since both items are on the same branch, re-pinning is also a viable option because it does not cause an unwanted duplicate item to occur.

Target Folder Has Floating Share in Source View (Row 200)

Defaults to **Fail**. Optionally, you can select **Ignore**.

This difference type occurs when the item being processed by View Compare/Merge exists only in the source folder. Without the folder problem, this item might have resolved to one of several different difference types:

- Share
- Reverse Share
- Move
- A compound action that includes one of the above actions, for example **Move and Merge** or **Delete-and-Reverse Share**.

This situation produces a **Fail** condition because if the item were moved, shared, or reverse-shared in the target, a duplicate copy of the item would float back into the source folder. You may see **Share**, or another of the possible merge actions, in the **Default Action** column for this item. This is because without the floating folder problem, the item's default merge action would have been **Share**. Because of the floating folder problem, View Compare/Merge overlays the item's difference type with the **Target Folder Has Floating Share In Source View (Row 200)** difference type. View Compare/Merge elects to **Fail** rather than cause multiple copies of the item in the source folder.

For example, suppose a folder was shared from the parent view to the child view, set to **Branch on Change** in the child view, but allowed to float in the child view. Further suppose that a new file was added to the folder in the child and you are promoting that item to the parent view. Normally, this file's difference type would have been **New in Source** and View Compare/Merge's default action would be a **Reverse**

Share. However, in this case, performing that action would cause a duplicate copy of the item in the source folder. That is why View Compare/Merge uses **Target Folder Has Floating Share In Source View (Row 200)** in this case instead of **New In Source (Promote) (Row 510)**.

You can ignore this situation by changing the action for the item to **Ignore** and then commit the rest of the session. Or you can end the session, go to the folder in the source view, and manually change its configuration from floating to pinned. When you redo the View Compare/Merge session, the item that had the difference type **Target Folder Has Floating Share in Source View (Row 200)** will have a different difference type, such as **New In Source**. The item's default action will be one of the merge actions listed earlier, and there will be no **Fail** condition.

It might be wise to check the other references for this folder to see if they, too, are floating—even if they are in other views—just to be sure that items will not float into other views that are not involved in this operation.

New In Source (Rows 500 through 520)

This series of difference types all have similar names but very different consequences.

New In Source (Row 500)

Defaults to **Share**. Optionally, you can select **Ignore**.

This item exists in the source but has no match in the target view. If this is a Rebase operation, the operation is going in the same direction as the view tree: from parent to child. In a Replicate operation, that may not be the case. Nevertheless, in either operation, View Compare/Merge shares this item into the target view at its current location in the source and pins its configuration to the time of the View Compare/Merge session.

New In Source (Promote) (Row 510)

Defaults to **Reverse Share.** Optionally, you can select **Ignore** or **Share**.

This difference type only occurs in promote operations. The only difference between **New In Source (Row 500)** and **New In Source (Promote) (row 510)** is that the View Compare/Merge operation is going in the opposite direction as the view tree: from the child to the parent.

To be sure that the reference tree for the item goes in the same direction as the view tree, View Compare/Merge does a reverse share. Next, the item's configuration is pinned to the time of the View Compare/Merge operation.

Unchanged

When View Compare/Merge reports a status **Unchanged**, it does not necessarily mean that the items have **not** been changed since the last View Compare/Merge session.

For example, suppose you change an item in one location and the changes float to another location. If these two items are compared in a View Compare/Merge session, View Compare/Merge reports **Unchanged** because the two items are identical.

For example, suppose changes have floated to both the items being compared. and then the child item changes again and branches, View Compare/Merge perceives that the pair was the same and that then the child changed. It does not recognize that both have changed since the original share or last View Compare/Merge session.

- If changes have floated to both items and one is deleted, View Compare/Merge reports the deletion but not the changes.
- If changes have floated to both items and then one is moved, View Compare/Merge reports the move only.

The floating issue can affect many rows of the View Compare/Merge Action Decision table.

Moved

If the files in one view and the files in another view have a common ancestor but are in different locations, they are reported as moved by View Compare/Merge because a move appears to have taken place. However, that may not be the case. They may have been shared manually from one view to a different location in another view initially. No move has really occurred. What View Compare/Merge is actually reporting is that they have different locations.

Suppose that all files were shared to a folder other than the corresponding folder in the other view, and were then moved to the corresponding folder.

- If you share to a folder other than the same parent folder view location, View Compare/Merge returns a **Moved** identifier, even though the item has not been moved but is still in its original shared location.
- If you share to a folder other than the parent location, and then move the item to that parent location, View Compare/Merge does not register the move. Instead, it lists the item as unchanged.

For example, in **Moved In Source (Row 1900)** promote:

- `Share2Execution` was shared from the `Execution0` folder in View 1 to the `Moveto` folder in View 2, then moved to the `Execution0` folder in View 2. It returned **Unchanged**.
- `SharerMoveTo` was shared from the `Execution0` folder in View 1 to the `Execution0` folder in View 2, then moved to the `MoveTo` folder in View 2. It returned **Moved in Source**.
- `Test3` was shared from the `Execution0` folder in View 1 to the `Moveto` folder in View 2 and not moved. It returned **Moved in Source** and the sub-folder was also listed again as **Unchanged**.

How View Compare/Merge Handles Target Folders with Floating Shares

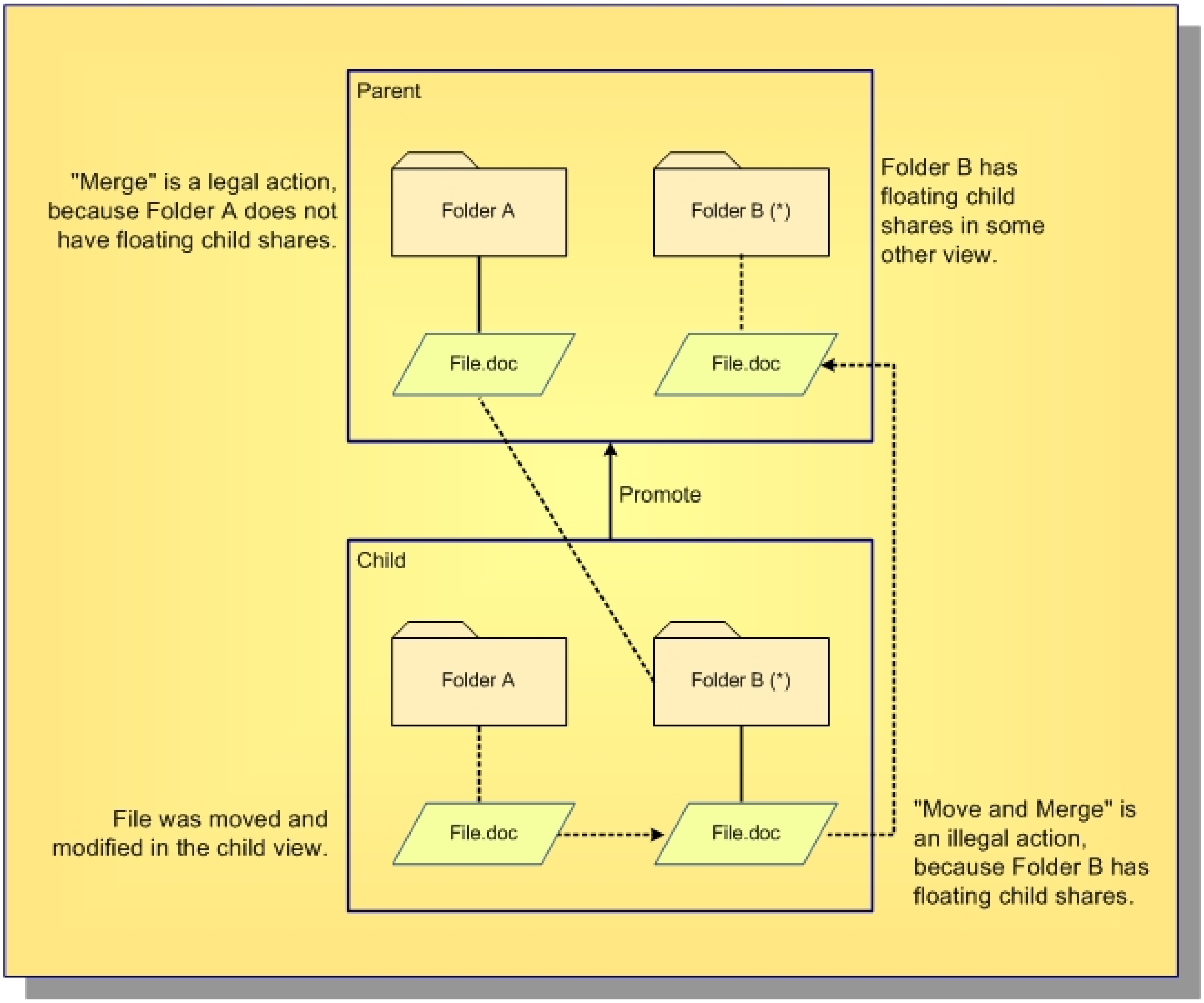
View Compare/Merge, by its very nature, tends to perform many move, share, and reverse-share operations. If one or more of the target folders has floating child shares elsewhere in the repository, and if View Compare/Merge performs a share, reverse-share, or move operation, View Compare/Merge could appear to be creating extra copies of items during the commit. Items could be automatically propagated to other locations throughout the repository, often creating floating child shares unintentionally. In such cases, subsequent operations that trigger automatic propagation can cause unexpected results.

To prevent this from happening, View Compare/Merge is defensive in the presence of folders that have floating child shares. It recognizes situations where an item is being shared, reverse-shared, or moved into a target folder that has active floating child shares, and disallows the operation reporting a *Fail*. To resolve the situation, you can review each failed difference and change the action (for example, to `Ignore`) before committing the session, or correct the floating share problem and redo the session.

Note: When matching an item for which there are multiple shares, View Compare/Merge prefers matches whose parent/child share relationship is the same as the parent/child relationship between the source and target views.

Example Move Operation in View Compare/Merge

Move operations present an interesting problem for View Compare/Merge. Consider the example below.


Parent
Folder A
Folder B (*)
File.doc
File.doc
Promote
Child
Folder A
Folder B (*)
File.doc
File.doc
"Merge" is a legal action, because Folder A does not have floating child shares.
Folder B has floating child shares in some other view.
File was moved and modified in the child view.
"Move and Merge" is an illegal action, because Folder B has floating child shares.

In this example, an item has been moved and modified in a child view. View Compare/Merge is then used to promote the changes to the parent view.

Suppose the user were to select **Move and Merge** as the current action for `File.doc`. In this case, the selected action would move the item into `Folder B`, which has floating child shares elsewhere in the repository. View Compare/Merge must therefore prevent the move operation from occurring.

On the other hand, if the user were to set the action to Merge, then the item would remain in its existing folder, and the operation would be permitted. Thus, View Compare/Merge must recognize the possibility that certain operations may change the target folder, and enable and disable actions appropriately as the target folder changes.

Target Folder Has Floating Shares

The View Compare/Merge engine has a difference type called “Target folder has floating shares” which is unlike the other difference types in that it is never selected from the View Compare/Merge Action Decision Table based on the `MatchState` that describes the relationship between a source item and target item. Instead, it is overlaid on top of an existing `DifferenceType` in the context of a given `DifferenceType`, based on an analysis of the target folder that would result from the currently selected action.

Consider again the example above. Use the View Compare/Merge Action Decision Table rows as the reference, where row [2] is “Target folder has floating shares”, and row [30] is “Moved and modified in source, target on different branch". In this case, the difference type selected from the decision table for `File.doc` is:

```
Row [30]: Moved and modified in source, target on different branch
    Action: Merge
    Legal Actions: Merge; Ignore; Move; Mark Resolved; Move and Merge; 
Overwrite
```

When a client application requests the difference type for this `ItemDifference`, the View Compare/ Merge engine looks at the currently selected action, and determines whether or not a move, share or reverse share will be performed. At this point, the current action is *Merge*, and therefore nothing special happens.

Now, suppose the user sets the action to *Move and Merge*. In this case, when the client application requests the difference type for this `ItemDifference`, the View Compare/Merge engine notes the move operation and determines whether or not the new target folder has any floating child shares. If it does, then the engine does not return the actual difference type, but instead returns "Target folder has floating shares".

Furthermore, the View Compare/Merge engine presents a different current action, and a different set of legal actions. These are determined dynamically, by removing all of the problematic actions and replacing them with Fail. Thus, by overlaying row [2] over row [30], the engine exposes the difference type to client applications as if it were the following:

```
Row [2]: Target folder has floating child shares

    Action: Fail
    Legal Actions: Fail; Merge; Ignore; Mark Resolved; Overwrite
```

Suppose now that the user changes the current action back to *Merge*. At this point, the engine no longer needs to overlay row [2] over row [30]; it exposes the difference type, current action and set of legal actions in their original, unaltered forms.

Frequency of Checking for Floating Shares

You should note that there is no way to prevent you from adding new floating shares. Thus, no matter when View Compare/Merge checks, it is always possible that a new floating share is added immediately afterwards, and is missed.

Checking often for a floating child share would decrease the likelihood that we would miss one. However, since this test is lengthy and resource intensive, this would result in unacceptable performance.

Currently, View Compare/Merge only checks for floating child shares in the source view since the source view is already open, and since checking the source view can be done rather easily and quickly.

To improve performance, View Compare/Merge checks each target folder at most once per session. However, for sessions that are saved and then later restored, View Compare/Merge checks again when the session is restored.

View Compare/Merge Process

This list of procedures presents the basic high-level tasks involved in performing a view compare/merge in StarTeam using View Compare/Merge.

1. Prepare your files for a View Compare/Merge session.
2. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
3. Resolve merge conflicts between the views.
4. Change any actions proposed by View Compare/Merge that you consider inappropriate for this session.
5. Test the merge changes to make sure they are all correct.
6. Generate a View Compare/Merge session report of all the merge changes.
7. Commit the merge changes.

Note: If you need to stop work on the change session before resolving and committing the changes, or you want to finish it later after having someone else review the proposed changes, you can use one of the **VCM Session** menu items **Save**, **Save to Local**, or **Export** before committing the session. StarTeam creates a *Change Package* object with a unique name in the target view which contains all the changes proposed for that change session. It is then available in the **Change Perspective** where it can later be restarted, reviewed, committed, and once committed, have the session Replayed which reuses the change package session to apply the same set of changes to another view.

Preparing Your Files for a View Compare/Merge Session

Before starting a View Compare/Merge session, take the following steps to prepare your files.

Note: To compare or merge specific items in a view, or all items in specific folders in the view, start the View Compare/Merge session from the **Advanced** context menu on the selected items, rather than from the **View** menu. When you start a View Compare/Merge session from the **View** menu, all items of requested types and folders in the view are included.

1. Check in any modified files that will be part of the session, and optionally create a view or revision label to use for the source.
2. Select the views you want to compare and set their properties to permit items to branch on change.

   Note: The target view is the only view that receives the results of the merge.
3. Decide whether you want to merge all changes in a view, or merge changes to selected items in a view.

   Select individual items in the upper pane if you are including specific items in the View Compare/Merge session. You can select multiple items on multiple component tabs, including the **Folder** tab.

Comparing and Merging Views

Use the **Compare Perspective** and the **Merge Perspective** in a View Compare/Merge session (View Compare/Merge session) to compare or merge the contents of a source and target view, and to review or change the proposed merge actions for resolving item differences.

Note: The steps and options included in the **View Compare/Merge Wizard** differ depending on which compare or merge type you select on the first page of the **View Compare/Merge Wizard**.

1. Choose **Project** > **Open** . The **Open Project Wizard** opens.
2. Select the project and one of the views you want to compare and click **Finish**.
3. In the opened view, select the root folder in the folder tree to compare items in the entire view, or select one of the sub folders to just compare items from that folder.
4. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
5. Select the type of compare or merge to perform: **Compare only**, **Promote**, **Rebase**, or **Replicate**.

   The **View Compare/Merge Wizard** uses the configuration to which the view is currently set to determine which choices to present on the first page of the wizard.

   Note: If you chose to perform a **Compare Only** merge of two reference views, on the **Select Source/Target Configuration** step of the **View Compare/Merge Wizard**, select **Current Configuration**. Do not select any of the other types of configurations for the **Compare Only** merge.
6. Click **Next** and continue through the wizard selecting the options you want until you get to the **Set Options** page of the wizard.

   Optionally, uncheck the **Start With Merge Perspective Current** checkbox option on the **Set Options** page of the wizard if you want to start the session in the **Compare Perspective**.

   This option is set to **Start With Merge Perspective Current** by default which causes the View Compare/Merge session to open in the **Merge Perspective**.

Select any other options you want and click **Next**.

Note: You can change perspectives as needed in the View Compare/Merge session to accomplish different tasks.

7. The **Exclude Properties** page appears next and gives you the opportunity to exclude any properties for each brancheable item type for which you do not want to merge changes when the session is committed.

   Select an item type in the **Item Type** list, then select a property for that item in the **Available properties** list that you want to exclude from the commit. Click the **Exclude** button to move it to the **Exclude properties** list.

   Note: If you change your mind, you can always select the item again, select the excluded property, and click **Include** to return it to the **Available Properties** list.
8. Click **Finish**. The View Compare/Merge session opens in either the **Merge Perspective** or **Compare Perspective**.
9. Use the **Merge Perspective** or **Compare Perspective** to change merge actions and resolve item differences.

   Tip: You can switch to the **Test Perspective** to see what the target view would look like if you applied your changes. Switch back to the **Merge Perspective** or **Compare Perspective** to make additional changes as needed.

Note: If you need to stop work on the change session before resolving and committing the changes, or you want to finish it later after having someone else review the proposed changes, you can use one of the **VCM Session** menu items **Save**, **Save to Local**, or **Export** before committing the session. StarTeam creates a *Change Package* object with a unique name in the target view which contains all the changes proposed for that change session. It is then available in the **Change Perspective** where it can later be restarted, reviewed, committed, and once committed, have the session Replayed which reuses the change package session to apply the same set of changes to another view.

Resolving Merge Conflicts in View Compare/Merge

You can resolve merge conflicts in the **Merge Perspective** or the **Compare Perspective** of a View Compare/Merge session. You can also open File Compare/Merge directly from the View Compare/Merge session to resolve differences in text file contents.

When merging views, no **Merge** or **Mark As Resolved** actions are possible for non-branching items, which are requirements, tasks, and topics.

1. Choose **VCM Session** > **Merge Perspective** or **VCM Session** > **Compare Perspective** to switch to one of these two perspectives.
2. Double-click an item with an `Unresolved`, `Resolved`, or `Merge` status.

   Note: The item status displays in the **Merge Action** column of the **Merge Perspective**, or in the **Action** column of the **Compare Perspective**.

   The **Resolve/Merge Properties** dialog box opens displaying the item properties.
3. Select a property with a conflict to resolve. Properties with conflicts display in **bold**.
4. Click **Use Source**, **Use Target,** or **Merge**.

   Tip: To quickly specify an action, double-click the **Source** or **Target** column for the property to use that value in the **Merged** column.
5. Optionally, double-click the **Merged** column on a text file to resolve the text differences manually.

When an unresolved item is resolved, View Compare/Merge updates the status to **Resolved** and changes the status icon to a check mark.

Tip: You can also resolve conflicts directly in the **Property** pane of the **Merge Perspective**. Right-click a property and choose an action from the context menu.

Note: If you need to stop work on the change session before resolving and committing the changes, or you want to finish it later after having someone else review the proposed changes, you can use one of the **VCM Session** menu items **Save**, **Save to Local**, or **Export** before committing the session. StarTeam creates a *Change Package* object with a unique name in the target view which contains all the changes proposed for that change session. It is then available in the **Change Perspective** where it can later be restarted, reviewed, committed, and once committed, have the session Replayed which reuses the change package session to apply the same set of changes to another view.

Changing View Compare/Merge Actions

This procedure describes how to resolve item conflicts in a View Compare/Merge session by changing the action you want applied to the target item. You can change merge actions for items with a status of **Resolved** or **Unresolved**.

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
2. Select an item in the upper pane of the **Merge Perspective** that has a status of **Unresolved** or **Resolved**.
3. Right-click the selected item and choose an action.

   Tip: You can also change the action by using the menu available when you hover the mouse over an icon in the upper pane.

   When an unresolved item becomes resolved, View Compare/Merge changes the status to **Resolved**. When all conflicts are resolved, you can commit your changes to the target view.

To view the current merge action for all items in the **Merge Perspective**, display the **Merge Action** column in the upper pane. You can sort the **Merge Action** column, and change the current action for one or more selected items using the context menu or flyout menu.

Testing View Compare/Merge Changes

1. Verify that you have resolved all merge conflicts in the **Compare Perspective** or **Merge Perspective** of the View Compare/Merge session.

   Note: You cannot check out individual files with merge conflicts in the **Test Perspective** of a View Compare/Merge session. Resolve all conflicts before using the **Test Perspective**.
2. Choose **VCM Session** > **Test Perspective** and check out the files in your project, including those with merged changes.

   View Compare/Merge saves all files locally to a temporary default working folder until you commit the View Compare/Merge session.

   Tip: You can specify an alternate View Compare/Merge working folder. Choose **VCM Session** > **Working Folder** in the **Test Perspective**.
3. Use your build tools to test that the project will build, run, and has the needed changes.

   Note: You cannot build or run from within StarTeam.
4. Optionally, switch to the **Merge Perspective** or the **Compare Perspective** and make changes as needed.
5. Recheck out your project if you have made additional changes, and rerun your tests.
6. Choose **VCM Session** > **Commit Changes** when you are satisfied with all the proposed changes.

   View Compare/Merge applies all the changes to the StarTeam Server repository and closes the session.

Viewing and Printing a View Compare/Merge Difference Report

At any time during a View Compare/Merge session, you can view a report of the differences in the session to see what you have done and what you have left to do. There are two types of difference reports available:

Difference Report	Displays information about the View Compare/Merge session parameters and options, a summary of the number of differences by type and action, and the view differences between the source and target by file and folder.
Detailed Difference Report	Displays the same information as the **Difference Report,** but also includes information about items whose parent folder has moved in the source or target view. If no folders were moved, this report will look the same as the **Difference Report**.

1. Start a View Compare/Merge session, or open a saved change package from the **Change Perspective** using **Change Package** > **Open (in VCM)** , or **Change Package** > **Open as Read Only (in VCM)** .
2. Choose **VCM Session** > **Difference Report** or **VCM Session** > **Detailed Difference Report** . The **VCM Session Differences** dialog box displays showing a two-column list of the session differences.
3. Optionally, click the **Browse** button to choose a different path and name for the output filename. The report is saved as an `.html` file.
4. Press the **View In Browser** button to display the difference report in your default browser.
5. Use the default browser menus to print the report.

Phases in the View Compare/Merge Process

View Compare/Merge provides consistent view compare/merge features across all client platforms. It supports refactoring and directionality. View Compare/Merge is optimized for distributed teams, and is oriented around merge tasks to facilitate the way you really work. It also allows the review and commit phases to be divided across time.

All sessions follow the same basic process which is divided into the following phases:

1. Definition
2. Comparison
3. Review
4. Save and Restore
5. Commit

Definition Phase

The View Compare/Merge parameters are specified for the compare phase. The session is defined in terms of its merge type, the source view (and snapshot), the target view, the source scope, and various options.

- In the StarTeam Cross-Platform Client, a session is started from a **View Compare/Merge** menu option which is available in several places, and which launches the **View Compare/Merge Wizard**. The context from which the wizard is launched affects the definition of the session.
- In the `VCMUtility`, the session is defined by command-line arguments and/or an options file.
- Both the StarTeam Cross-Platform Client and `VCMUtility` provide a variety of ways to define the source scope items. The source scope can be:
  - The file revisions linked to one or more process items (and optionally the process items themselves).
  - The items attached to a revision label.
  - Items of specific types that reside in selected folders.
  - A set of explicitly selected items; and items of selected types in the entire view.

The definition phase is where you define the process for the comparison. In this phase, you can:

- Specify the view merge type.
- Specify the source view, item(s), target view, and options.
- Specify the type of items to include.

Comparison Phase

The Comparison phase is where the comparison takes place. The comparison takes place automatically when you finish the wizard before the View Compare/Merge opens in the StarTeam Cross-Platform Client.

Once a session starts, View Compare/Merge first performs the Comparison phase.

- "In scope" source items are matched to corresponding items in the target view. View Compare/Merge detects both simple changes (items that are new, modified, or deleted in one or both views) and more complex changes (items that have been moved, renamed, modified-and-moved, and so on.)
- For an item that exists in both the source and target view, it determines if either item or both have changed since the last time they were merged.
- For each difference found, it defines an *item difference* that specifies the source and/or target item (sometimes there is only one of these) as well as a default *action*. The action specifies what the session will do about the item difference. Many actions are possible including `ignore`, `move`, `merge`, `delete`, `reverse share`, `move and merge`, and so on.

Review Phase

After the comparison phase has completed, the results can be reviewed, adjusted, and validated before the session is committed. The review phase is where you analyze your comparison and make any necessary adjustments.

In the Review phase of a session, you can

- Preview what the target view would look like if the session was committed in its current state. The StarTeam Cross-Platform Client provides this preview as a **Test Perspective**. The `VCMUtility` provides a `CheckoutPreview` option that allows you to check out files from the preview.
- Generate a difference report in both the StarTeam Cross-Platform Client and `VCMUtility` that summarizes all item differences and their current actions.
- Resolve differences requiring user input, and manually merge file contents and property values when there are conflicts. Actions are considered resolved if no user input is required to continue. However, unresolved actions such as differences with merge conflicts, require user input. The session cannot be committed without user attention. As a best practice, all actions should be reviewed and adjusted if necessary.
- Choose from multiple possible actions to resolve differences where applicable. For example, for an item in a merge state (modified in both views since the last compare), you could choose to overwrite the target item to match the source item instead of choosing to merge the two.
- Check out files to working folders, perform a build, and run tests on the simulated merge.
- Define properties of the change package that will be created when the session is saved or committed such as its name and description.

Save and Restore Phase

Optionally, at any time after the compare phase, you can save the session and restore it later. By default, a session is saved as a change package in the target view identified by the session. A session can also be saved as a local session file or exported as a self-contained exchange file. All three ways of saving a session allow it to be restored later. Restoring a session "resumes" the session at the same point at which it was saved.

- When a session is saved as a change package, you can copy a URL for it to the clipboard and paste it into an e-mail, for example, so others use the URL to open the change package.

- A session saved as a Session file (`.vcms` file extension) is a shortcut that can only be used to restore the session on the same computer. This is because temporary files are saved on the workstation required to resume the session.
- A session exported as an Exchange file (`.vcmx` file extension) is a self-contained “change package” file that can be copied to another computer, e-mailed, checked into StarTeam or added to a change request as an attachment, and so on. Others can then restore, review, and even commit the session.

Change packages are the preferred method for saving and restoring sessions.

Commit Phase

In the commit phase, View Compare/Merge applies changes to the target view based on the decisions made during the compare and review phases, and records the changes in the StarTeam repository as a change package.

Based on session options, various labels can be automatically created that document what changed in the target view. View labels can be created that provide a snapshot of the target view before and/or after the updates were applied. Revision labels can be created that are attached to target view items affected by the session before and/or after the updates were applied.

View Compare/Merge can also generate a post-commit update report, which enumerates exactly what was changed by the change session.

View Compare/Merge Wizard

This section describes the individual pages of the **View Compare/Merge Wizard**. The order in which the pages occur in the wizard depends on the type of comparison you are performing and from which view you start the wizard.

Each step of the **View Compare/Merge Wizard** contains a button at the bottom right the page which, when clicked, opens an information pane that shows you the details of the choices you have made for the session up to that step in the **View Compare/Merge Wizard**. To hide this information pane, click the button again and it will disappear. The advantage of this information pane is that you can review in one place all the details of the session before you click **Finish** without having to go back and forth through the different pages of the wizard to see what you selected. That way, if you decide you want to make a different choice, you can go back to a specific page of the wizard, make any changes, then click **Finish** at that point.

View Compare/Merge Wizard: Select Compare/Merge Type

Use the this page of the **View Compare/Merge Wizard** to select the type of compare/merge action to perform.

Select The Type Of Compare Or Merge	Specifies the type of compare merge to perform:	
	Compare only	Compare changes between one related view and another, or between two different configurations of the same view. (Default).
	Promote	Compare or merge changes from a child view to its parent.
	Rebase	Compare or merge changes from a parent to its child view.
	Replicate	Compare or merge changes from one related view to another.
Use Current View (name) For:	Specifies whether the current view should be used as the source view or the target view, or as both the source and target (compare only). If you choose **Promote** or **Rebase** as the merge type, the wizard may automatically select the appropriate option based on whether the parent view or a child view is the current view.	

View Compare/Merge Wizard: Include Items

Use this page to limit the compare/merge to specific types of items. This step only appears if you start the **View Compare/Merge Wizard** from the **View** menu.

Revision Label Compares items with the selected revision label only. When checked, the list is enabled and displays the available revision labels.

Include Specifies which types of items in the view to include in the compare/merge.

View Compare/Merge Wizard: Include Selected Items

Use this page to limit the compare/merge to specific types of items. This step of the wizard only appears if you right-click specific items on the component tabs in the upper pane choose **Advanced View Compare/Merge** on the context menu to start the compare/merge session.

Include Specifies which types of items to include in the compare/merge.

The **Include** tree displays an enabled check box for each item type you have selected, and also gives you the option of selecting all files linked to the selected item type if it is a process item. When you check **Selected Folders** in the **Include** tree, you can also check **Subfolders** to include all contained child folders and their children in the compare.

Selecting a folder from the **Folder** component tab instead of the in the **Folder Tree** enables you to click other item tabs and select additional items from other folders to include in the compare/merge.

View Compare/Merge Wizard: Select Source View

Use this page of the wizard to select the source view for the compare/merge.

View Tree Uses the selected view as the source view for the compare/merge. Click to select.

View Name Displays the name of the view selected in the **View Tree**.

View Description Displays the description of the selected view. A view description is entered in the **Description** field of the **View Properties** dialog box.

View Compare/Merge Wizard: Select Target View

Use this page of the wizard to select the target view for the compare/merge.

View Tree Uses the selected view as the target view for the compare/merge. Click to select.

View Name Displays the name of the view selected in the **View Tree**.

View Description Displays the description of the selected view. A view description is entered in the **Description** field of the **View Properties** dialog box.

View Compare/Merge Wizard: Select Source Configuration

Use this page of the **View Compare/Merge Wizard** to restrict the source view to a specific configuration in the View Compare/Merge session.

Source View Configuration for (view name) Specifies a configuration to use for the View Compare/Merge session so you can isolate the view compare/merge from changes in the current view. Select from:

Current Configuration This uses the view's current configuration.

Labeled Configuration This uses the configuration with the revision label you specify.

Promotion State Configuration This uses the configuration with the selected promotion state.

Configuration As Of This uses the configuration from the specified date and time.

View Compare/Merge Wizard: Select Target Configuration

Use this page of the wizard to restrict the target view to a specific configuration in the View Compare/Merge session.

Target View Configuration for (view name) Specifies a configuration to use for the View Compare/Merge session so you can isolate the view compare/merge from changes in the current view. Select from:

Current Configuration This uses the view's current configuration.

Labeled Configuration This uses the configuration with the revision label you specify.

Promotion State Configuration This uses the configuration with the selected promotion state.

Configuration As Of This uses the configuration from the specified date and time.

View Compare/Merge Wizard: Set Options

This page of the **View Compare/Merge Wizard** contains options that will control the beginning of the View Compare/Merge session. You can change individual default actions in either the **Compare Perspective** or the **Merge Perspective** during the View Compare/Merge session.

Auto-merge Properties View Compare/Merge automatically merges the view, folder, or item properties at the beginning of the View Compare/Merge session. For example, if a property has changed only in the source, the value of that property in the target is changed to match. (Default)

Auto-merge Files View Compare/Merge automatically merges files without conflicting differences at the beginning of the View Compare/Merge session. (Default)

Match Files By Name View Compare/Merge matches files with the same file names, even if the object IDs are different, or if the files cannot otherwise be matched. For example, if the same line has changed in both a source and target text file, View Compare/Merge cannot automatically merge the two lines, but if the line has changed only in the source, those changes can also be made to that line in the target. When you match source and target files by name, and the object IDs are different, no branch point can be used to locate a common ancestor when merging or comparing the two files. However, if you have compared these two files (in this source-target direction) before, there may be merge points. (Default)

Treat File Names As Case Sensitive If **Match Files By Name** is checked, file matching is case sensitive.

Ignore Merge Points If checked, View Compare/Merge ignores existing merge points by default. If unchecked, View Compare/Merge uses the last recorded merge point between two items to determine if either or both items has changed. When View Compare/Merge performs a merge on a text file, the revision of one of the files (usually the source file) is used as the least common ancestor to perform the merge. When merge point usage is not enabled, the least common ancestor is determined by finding the last branch point common to the two files. It should be very rare that you would ever want to ignore merge points.

Fix/Find Floating Child Shares

Fixes any shares that are floating by pinning them in a Rebase or Promote, since View Compare/Merge cannot merge floating shares. In a Compare-only session, VCM simply finds the floating child shares for you.

In Rebase and Replicate merge operations, this option specifies whether each target view item found that is a floating share of a source view item should be “fixed” by pinning it. When a target view item is a floating child share of a source item (which implies that the target item has not branched), differences will not be detected between the source and target item during VCM sessions because changes to the source item immediately float to the child item. View Compare/Merge best practices suggests that child shares should always be pinned, allowing changes to propagate from the source to target view in a controlled manner. This option allows floating child items found by View Compare/Merge to be “fixed” by pinning them to the parent item revision. Specifying this option has a performance cost due to the extra commands required to check each target item examined during the compare phase.

Tip: You might want to check this the first time you perform a compare/merge on a view, then uncheck it for successive comparisons to speed up your sessions.

Break Locks Automatically

Attempts automatically to break an existing lock if the item is already locked by another user, and if the item needs to be locked due to one of the locking options below being set. Breaking locks requires special permissions. **Break Locks Automatically** is only relevant when View Compare/Merge wants to lock an item. And it only locks items for actions that make changes. For example, if you are in the middle of a session, and you change an action from **Ignore** to **Delete**, View Compare/Merge may break someone else's lock on the item that is going to be deleted at commit time.

Lock Source For Difference

Locks source items for differences that will result in a change. Some actions, such as **Ignore**, will not result in a change, and therefore do not require a lock. Locking usually happens at compare time, but may also happen later in the session if the action for a difference is overridden. View Compare/Merge will not lock reverse-shared items in the source view unless you check this option in the **View Compare/Merge Wizard**.

Lock Target For Difference

Locks target items for differences that will result in a change. Some actions, such as **Ignore**, will not result in a change, and therefore do not require a lock. Locking usually happens at compare time, but may also happen later in the session if the action for a difference is overridden. (Default)

Start With Merge Perspective Current

Starts the View Compare/Merge session in the **Merge Perspective**. If unchecked, View Compare/Merge starts in the **Compare Perspective**. (Default)

View Compare/Merge Wizard: Exclude Properties

Use this page to exclude properties of any branchable item type.

Item type

Provides a list of the selected item types this session whose properties you can exclude from a View Compare/Merge session. The item types in this list are determined by the choices of which item types you selected for View Compare/Merge in the **View Compare/Merge Wizard**.

Available properties Lists all the properties available for the Item type selected. Select an item in the **Available properties** list and click the **Exclude** button to move it from **Available properties** to the **Exclude properties** list.

Exclude properties Lists all the properties selected for exclusion for the selected Item type. You can move an item from the **Exclude properties** list back to the **Available properties** list if you change your mind. Select the item in the **Exclude properties** list and click the **Include** button to move it back to the **Available properties** list.

Finding and Breaking Locks on Items with Differences

If an item with differences is locked by another user in the target view for the merge, you will be unable to commit changes for that item in the View Compare/Merge session unless you break the lock. When you start a View Compare/Merge session, the **View Compare/Merge Wizard** can find the locks and break them automatically if you check that option in the wizard. However, you can also break the locks manually in the session.

Caution: Items are typically locked with exclusive locks because they are being edited. If you break another user's lock, you may cause that user to lose work. You should break another user's locks only after contacting them and in urgent cases where they cannot be contacted.

Breaking Locks Automatically When Starting a Session

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
2. On the **Set Options** page, check **Break Locks Automatically**.
3. Check any other options you want, then click **Finish** to complete the wizard.

Finding Locked Items with Differences

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
2. In any perspective, choose **Difference** > **Find Next Locked** , or **Difference** > **Find Previous Locked** .

Breaking Existing Locks in a Session

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
2. In any perspective, choose **Difference** > **Find Next Locked** , or **Difference** > **Find Previous Locked** .
3. Right-click the item in the upper pane and choose **Lock/Unlock**. The **Set My Lock Status** dialog box opens.
4. Select the **Unlocked** lock status option to remove the exclusive or non-exclusive lock.
5. Check **Break Existing Lock** to break another team member's lock on the item.

If e-mail is enabled, StarTeam will send an e-mail message to the team member whose lock has been broken to inform him or her of this fact.

Note: You must be granted the appropriate privileges to be able to break another user's locks.

Skipping a Revision in VCM

When you run a View Compare/Merge session, if the default action on a file is *RE-PIN* or *MERGE*, there is a new menu option called **Skip a Revision**.

If you select this item, you get a dialog box of all historical revisions of the file to pick from.

When you select a revision, View Compare/Merge attempts to temporarily remove that revision content from the source file, then use this output to regenerate a merge to the target file.

Comparing a Source and Target File in View Compare/Merge

In View Compare/Merge, both the **Compare Perspective** and the **Merge Perspective** compare the contents of views. The **Compare Perspective** compares and displays the contents of the source and target views, and provides a list of changes View Compare/Merge will make on commit. The **Merge Perspective** shows items with proposed merge actions, organizing them by folder and filter as in a regular view window. You can also view a comparison of the differences in individual file contents from the **Merge Perspective**. Both perspectives allow you to open StarTeam File Compare/Merge to resolve and merge the differences.

Note: When you select a file in the **Merge Perspective** and select the tab that compares the target file to the merged file, it is blank if there is no merged file. While the selected file is **Unresolved**, the tab is blank. If the user merges the file or selects a resolving action such as **Overwrite**, then the pane displays content comparing the target file to the content of the resolved file. In the case of the **Ignore** action, the target file is the resolved/merged file. In the case of the **Overwrite** action, the target file is the source file.

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**. By default, the View Compare/Merge window opens in the **Merge Perspective**. Stay in this perspective.
2. Select a file with differences, having a merge status of either **Unresolved** (question mark), or **Resolved** (check mark).
3. Click the **File Comparison: Source, Target** tab at the bottom.

You can only compare the file differences in View Compare/Merge. To merge the differences, you can open StarTeam File Compare/Merge.

Note: Color highlighting of differences in the **Compare** pane are explained and controlled in the **File Compare/Merge Options** dialog box. To change the default highlighting colors for file and folder comparison differences for both View Compare/Merge and StarTeam File Compare/Merge, modify the File Comparison and Folder Comparison Color Preferences accessed by choosing **Tools** > **Options** in the StarTeam File Compare/Merge stand-alone window.

Comparing Selected Items in the Source and Target View

1. Open the view to use in your session.
2. Choose **View** > **Configuration** .

   The **View Configuration** dialog box opens.
3. Choose a configuration unless you plan to use the current one.

   Note: The **View Compare/Merge Wizard** uses the configuration to which the view is currently set if you start the wizard from the source view.
4. Select the individual items to compare on the component tabs in the upper pane, or select a folder from the folder tree.

   Note: The items you select on one tab will stay selected when you switch component tabs.
5. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
6. Select the type of compare or merge to perform: **Compare only**, **Promote**, **Rebase**, or **Replicate**.
7. Click **Next** and continue through the wizard selecting the options you want.

Note: The remaining steps and options in the wizard differ depending on which compare or merge type you selected.

8. On the **Set Options** page of the wizard, uncheck **Start With Merge Perspective Current**.

   Leaving this option unchecked causes the View Compare/Merge window to open in the **Compare Perspective**.

The upper pane displays the contents of the source and target views. It highlights items with no matches in green, matched items with differences in yellow, and items moved to another folder in blue. You open, close, and navigate folders. Each comparison pane contains scroll bars and navigation buttons. Selecting an item in the top panes also selects it in the lower pane. Use filters to control which items display in the upper pane.

The lower pane displays information about the items in the upper pane. You can modify the merge action for each item and resolve differences.

Comparing a Target File with the Proposed Merge Results

When you are in a View Compare/Merge session, you can compare the text contents of the target file with the proposed merge results using the **Merge Perspective**.

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**. By default, the View Compare/Merge window opens in the **Merge Perspective**. Stay in this perspective.
2. Select a file with differences, having a merge status of either **Unresolved** (question mark), or **Resolved** (check mark).
3. Click the **File Comparison: Target, Merged** tab at the bottom of the lower pane.

Note: This pane only allows you to compare the file differences. To merge the differences, you can open StarTeam File Compare/Merge.

Scoping a Session with a Process Item in a Different View

Use the following procedure to scope a session with a process item in a different view than your current process. This is useful if you are using cross-view or cross-project process items.

Note: Use the **All Change Packages by Process Item** filter in the change package perspective for sorting change packages by the process item used during commit.

1. Select the new active process item.
2. Click **View** > **Compare/Merge Active Process Item**. The **View Compare/Merge Wizard** opens.
3. Follow the **View Compare/Merge Wizard**, which by default will scope the View Compare/Merge session to the active process item and it's changes.

Merging Changes Between Any Two Related Views (Replicate)

A Replicate View Compare/Merge session compares or merges changes between any two views that are in the same project. You can start a Replicate View Compare/Merge session from either the source view or the target view. To perform a Replicate operation using the current view as the source or target, follow the steps below.

1. Open the view to use in your session.

2. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
3. Select **Replicate** as the compare/merge type.
4. Select **Source of Merge** or **Target of Merge** under **Use Current View (name) For**, and click **Next**.
5. If you selected **Source of Merge**, choose the source view configuration to use: **Current**, **Labeled**, **Promotion State**, or **Configuration As Of** (a specific timestamp). The **View Compare/Merge Wizard** uses the configuration to which the view is currently set if you start the wizard from the source view.
6. Select the target or source view in the **View Tree** for the comparison and click **Next**.
7. Select one or more types of items you want to include in the compare/merge session and click **Next**.
8. Check one or more compare/merge options to use by default at the beginning of the compare/merge, and click **Finish**.

View Compare/Merge begins comparing the views immediately and opens a new StarTeam window in the **Merge Perspective** where you can resolve any remaining merge conflicts. Items with unresolved differences are easy to find. In the **Merge Status** column for each of these items, View Compare/Merge displays a question mark, followed by the word **Unresolved**. The component tabs and folders also display a question mark if they contain items with unresolved differences.

You must resolve each merge conflict before you can complete the merge and commit the changes.

Merging Changes from a Child View to a Parent View (Promote)

Promote compares or merges changes from a child, or derived, view to its parent. You can start a Promote View Compare/Merge session from either a parent view or a child view. To perform a Promote operation starting from the parent or child view, follow the steps below.

1. Open the view to use in your session.

   Note: The parent is the target in a Promote operation and its tip configuration must be used. The **View Compare/Merge Wizard** uses the configuration to which the view is currently set if you start the wizard from the source view.
2. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
3. Select **Promote** as the compare/merge type.
4. Select **Source of Merge** or **Target of Merge** under **Use Current View (name) For**, and click **Next**.

   Note: Both the **Source of Merge** and **Target of Merge** options can be enabled if the parent or child view is both the parent and child of another view.
5. Select the parent or child view to use as the source view in the **View Tree** for the comparison and click **Next**.
6. If you selected **Target of Merge**, choose the source view configuration to use: **Current**, **Labeled**, **Promotion State**, or **Configuration As Of** (a specific timestamp).
7. Click **Next**.
8. Select one or more types of items you want to include in the compare/merge session and click **Next**.
9. Check one or more compare/merge options to use by default at the beginning of the compare/merge, and click **Finish**.

View Compare/Merge begins comparing the views immediately and opens a new StarTeam window in the **Merge Perspective** where you can resolve any remaining merge conflicts. Items with unresolved differences are easy to find. In the **Merge Status** column for each of these items, View Compare/Merge displays a question mark, followed by the word **Unresolved**. The component tabs and folders also display a question mark if they contain items with unresolved differences.

You must resolve each merge conflict before you can complete the merge and commit the changes.

Merging Changes from a Parent View to a Child View (Rebase)

Rebase compares or merges changes from a parent view to one of its child, or derived, views. You can start a Rebase View Compare/Merge session from either a parent view or a child view. To perform a Rebase starting from the parent or child view, follow the steps below.

1. Open the view to use in your session.

   Note: The **View Compare/Merge Wizard** uses the configuration to which the view is currently set if you start the wizard from the source view. Use the tip revision in the child view because the child is the target in a Rebase operation.
2. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
3. Select **Rebase** as the compare/merge type.
4. Select **Source of Merge** or **Target of Merge** under **Use Current View (name) For**, and click **Next**.

   Note: Both the **Source of Merge** and **Target of Merge** options can be enabled if the parent or child view is both the parent and child of another view.
5. If you selected **Source of Merge**, choose the source view configuration to use: **Current**, **Labeled**, **Promotion State**, or **Configuration As Of** (a specific timestamp).
6. Click **Next**.
7. Select the child or parent view to use as the target view in the **View Tree** for the comparison and click **Next**.
8. Select one or more types of items you want to include in the compare/merge session and click **Next**.
9. Check one or more compare/merge options to use by default at the beginning of the compare/merge, and click **Finish**.

View Compare/Merge begins comparing the views immediately and opens a new StarTeam window in the **Merge Perspective** where you can resolve any remaining merge conflicts. Items with unresolved differences are easy to find. In the **Merge Status** column for each of these items, View Compare/Merge displays a question mark, followed by the word **Unresolved**. The component tabs and folders also display a question mark if they contain items with unresolved differences.

You must resolve each merge conflict before you can complete the merge and commit the changes.

Using Process Items to Merge Related Files

View Compare/Merge allows you to compare and merge one or more sets of files related to specific process items by selecting the process items to which they are linked and pinned (instead of selecting the files themselves).

1. Choose **Project** > **Open** on the StarTeam client menu, and open the source view containing the process items you want to use.
2. Choose **View** > **Select Configuration** and specify the source view configuration you want to use by selecting the appropriate configuration type: **Current**, **Labeled**, **Promotion State**, or **Configuration As Of**.

   Note: The **View Compare/Merge Wizard** uses the configuration to which the view is currently set if you start the wizard from the source view.
3. Select the process items from the **Change Request**, **Requirement**, or **Task** tabs in the source view.

   Note: Use **Shift+Click** or **Ctrl+Click** to select multiple items in the upper pane. The items you select stay selected when you switch tabs or project windows.

4. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
5. Select the type of compare or merge to perform: **Compare only**, **Promote**, **Rebase**, or **Replicate**.
6. Select **Source Of Merge** under **Use Current View [name] For**, and click **Next**.
7. Check **Changes associated with [item]** on the **Include Selected Items** page of the wizard.
8. Check **Selected [item]** to compare and merge the selected items as well as the files linked to them.

   Caution: The link to each file must be pinned at both ends or it is not included.
9. Click **Next** and complete the wizard.

View Compare/Merge begins comparing the views immediately and opens a VCM session window in the **Merge Perspective** where you can resolve differences in the process items, and/or the files linked and pinned to them. Items with unresolved content or property differences are easy to find. In the **Merge Status** column for each of these items, View Compare/Merge displays a question mark, followed by the word **Unresolved**. The component tabs and folders also display a question mark if they contain items with unresolved differences.

If View Compare/Merge can propose an appropriate merge action, it displays the item's status as **Resolved** with a green check mark. You can change any proposed action with another possible action.

You must resolve each merge conflict before you can complete the merge and commit the changes.

Resolving File Differences from a View Compare/Merge Session

If you have content differences between two text files, you can open File Compare/Merge directly from the View Compare/Merge session to resolve the differences. You can also change the action from **Merge** to **Overwrite**, **Mark As Resolved**, or **Ignore**.

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**. By default, the View Compare/Merge window opens in the **Merge Perspective**. Stay in this perspective.
2. Select a file with differences (having a merge status of either **Resolved** or **Unresolved**).
3. Click the **Property** tab in the lower pane. The **<File Content>** property is displayed in the **Property** pane with boldface type since it has differences.
4. Right-click in the row for the **<File Content>** property and choose an action to resolve the conflict. Choose **Use Source** to use the version of the file in the source view, **Use Target** to use the version in the target view, or **Merge** to open File Compare/Merge to compare and merge the contents.

   Tip: To quickly specify a resolution, double-click the **Source** or **Target** column for the **<File Content>** property to use that value for the **Merged** column. You can also double-click the **Merged** column for the **<File Content>** property to open File Compare/Merge to merge the files.

Committing View Compare/Merge Change Session

Committing a View Compare/Merge change package session applies all changes to the StarTeam repository. You must resolve all merge conflicts before you can commit the View Compare/Merge change session.

1. Open the target view in the Cross-Platform Client **Content Perspective** containing the change session you want to commit.
2. Double-click the change package you want to commit. It will be listed in the list of change packages in the **Content Perspective** and will have a status of **New**.

   Note: If you do not see the session you want, make sure the **Filter** list has the correct selection so all sessions are displayed.

3. Double-click the change package session to open it in the session.
4. Verify that you have resolved all merge conflicts in the **Compare Perspective** and the **Merge Perspective** of a change session, and that you have tested your changes to make sure they give the expected results. When the changes in a change session are complete, it can be committed.
5. Choose **VCM Session** > **Commit Changes** .

   Note: If **Commit** is disabled on the menu or toolbar, check the **Compare** and **Merge Perspectives** to make sure all conflicts are resolved, and items with changes are not locked by other users, then try again.
6. Type a merge session comment in the **Commit Changes to View** dialog box.
7. Click **OK** in the **Commit Changes to View** dialog box to commit your View Compare/Merge session.

   This causes its updates to be applied to the target view in an atomic transaction. In the same transaction, the change package object is updated with details about updated items. After commit, the change session is closed and can no longer be opened for editing. In its place, the committed change package appears as an object owned by the target view, and can be viewed again in the **Change Perspective**.
8. Save your change session using **VCM Session** > **Save** , **VCM Session** > **Save to Local** , or **VCM Session** > **Export** .

   Note: Save saves the session persistently, minimizing the need to save the session as a file. **Save to Local** saves the session as a `.vcms` file, and **Export** saves the session as a `.vcmx` file.

StarTeam creates a uniquely named change package object in the target view for the change session which contains all the changes made in that VCM change session. It then becomes available in the **Change Perspective**, where it can later be opened in read-only view, reviewed, have the properties modified, or have the session Replayed which allows you to reuse the change package session to apply the same set of changes to another view. Once a change package is committed, no more changes can be made to it.

View Compare/Merge Optimization

View Compare/Merge optimization is a workaround for very large View Compare/Merge commits (that is, tens of thousands of View Compare/Merge operations), which cannot realistically succeed when run in a server and database transaction.

To use this workaround, the session must be committed by a user who has the **Modify system configuration** Server Access Rights granted and is preferably using the 64-bit SDK or StarTeam Cross-Platform Client. The session should also be run when there is little to no other activity on the StarTeam Server, because of the extensive resource allocation to be provided for this commit session.

1. Turn on the setting *AtomicBehavior* in the `starteam-server-configs.xml` file. There is no need to stop and restart the server just to change this setting.

   The default value of this setting is 0, meaning that Atomic Transactions are enabled by default.

   ```
   <option name="AtomicBehavior" value="0"/>
   ```
2. Set the *AtomicBehavior* setting value to 1.

   ```
   <option name="AtomicBehavior" value="1"/>
   ```
3. Wait for a minute, then run the large View Compare/Merge commit from a 64-bit version of the SDK or StarTeam Cross-Platform Client using the Administrative Context with privileges to modify server settings.

   Note: Running the session from a 32-bit SDK or StarTeam Cross-Platform Client runs the risk of the StarTeam Cross-Platform Client running out of memory before the session is committed.

When the session starts up and discovers that *AtomicBehavior* is disabled, it will immediately re-enable it and then run the session without transactions. This means that a subsequent sessions will automatically run with transactions enabled.

If the session is run using non-administrative privileges, then the attempt to re-enable transactions in the server will fail, and the setting will be ignored. In this case, the session will run with multi-command transactions enabled, as if the settings were unchanged.

Internal testing has indicated that a session with 65,000 files being reverse-shared run in 2 hours on a single machine and 5 hours across a network with this new workaround. Without the workaround (that is, with transactions enabled), the same commit takes over 24 hours and does not complete.

Exporting a View Compare/Merge Session

Before you leave a View Compare/Merge session, you can export it to a file so you can open the session again later, or transfer the session to a different computer. View Compare/Merge saves all the session information in a `<VCM exchange file>` file with a `.vcmx` extension to which you can send a StarTeam shortcut URL to others to open, or which can be sent to others for review, or opened from the **Change Perspective**.

Note: If you only expect to open the session later on the same machine, to reduce the file size, you can use **VCM Session** > **Save** , or **VCM Session** > **Save to Local** . The `.vcms` file created with a **Save to Local** only saves part of the session information. Data from the local repository directory is not included in this file. This format is used to automatically save data when closing StarTeam with a View Compare/Merge session open and only allows you to open it on the machine on which it was saved.

1. Choose **VCM Session** > **Export** on the View Compare/Merge menu.

   This opens the **Export VCM Session As** file browser.
2. Make sure that the **Files of Type** field has `VCM Session Files(*.vcmx)` selected.
3. Browse to the target location for the `.vcmx` file.
4. Optionally, give the file a different name. You can also specify the name and location for a shortcut file.

   Note: The default name consists of the type of compare/merge it is, followed by the date and time of the session.
5. Click **Save** to export the session to the `.vcmx` file.

Note: An exported View Compare/Merge change session will be displayed in the **Change Perspective** of the Cross-Platform Client. If it was uncommitted before it was saved, you can restart it in the **Change Perspective**, review and make any changes, then commit the change session.

Opening a Saved or Exported View Compare/Merge Session

You can save or export a View Compare/Merge session to a shortcut file (`.vcms` or `.vcmx`), and later restore the session for more work.

- If you save the session to a `.vcms` file, you can open it again on the same computer.
- If you export the session to a `.vcmx` file, you can move the file to a different computer and restore the session there.

Note: The `.vcmx` file contains more information than a `.vcms` file, such as merged result files.

1. Open the StarTeam Cross-Platform Client.
2. Choose **Project** > **Open Shortcut** . The **Open** dialog box displays.
3. Select `VCM Session (*.vcmx;*.vcms)` in the **Files of Type** field.
4. Browse to the location of the session file you want to restore and click **Open**.

Restarting a View Compare/Merge Session

This procedure describe how to restart an uncommitted View Compare/Merge session so you can start completely over with the same initial session. There are two ways to restart a View Compare/Merge session. Both result in the same outcome.

Restarting a View Compare/Merge session restores it to the same state it was in when you finished running the **View Compare/Merge Wizard** to originally start the session. If you previously resolved any conflicts before you restart the session, all those changes will disappear, and you will be returned to the beginning of the session where you can start over reviewing and editing changes.

1. In the StarTeam Cross-Platform Client, open the target view of the change package you want to use.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Choose the **New** (uncommitted) change package you want to restart.
4. Do one of the following:

   - In the **Change Perspective**, choose **Change Package** > **Restart Session** . The selected session will open in a session in its original state immediately after running the **View Compare/Merge Wizard**.
   - In the **Change Perspective**, choose **Change Package** > **Open (in VCM)**, or double-click the selected session. This opens the change package in the session as it was last saved with the most recent changes to the session. On the session menu, choose **VCM Session** > **Restart Session** . Now all the changes made are removed and the session returns to the state it was in right after running the **View Compare/Merge Wizard**.

The advantage of the **Restart Session** operation lets you completely start over without having to re-run the **View Compare/Merge Wizard**. This can save a lot of time for especially large sessions.

Viewing and Printing a View Compare/Merge Session Summary

At any time during a View Compare/Merge session, you can view the session summary.

1. Start a View Compare/Merge session using the **View Compare/Merge Wizard**.
2. Choose **VCM Session** > **Summary** . The **VCM Session Summary** dialog box opens and displays a two-column list of the session properties.
3. Optionally, click the **Browse** button to choose a different path and name for the output filename. The report is saved as an `.html` file to the specified location.
4. Optionally, click **View In Browser** to display/print the session summary in your default browser.

Deleting a View Compare/Merge Session

You can delete an active View Compare/Merge session which removes all session information from your computer.

1. Click **VCM Session** > **Delete** .
2. Click **Yes** or **No**.

   Yes deletes the active session in the client, the session shortcut file (`.vcms`) and the session repository folder. **No** closes the active session in the client, but does not delete the shortcut file or session repository.

Note: When you close a session normally, the session is lost unless it was previously saved, or you modify it and save it when closing.

Changing the Default Merge Type Options and Actions

As a server administrator you can modify the availability of default and custom merge types and options when users launch the **View Compare/Merge Wizard**.

1. Open the **Server Administration** tool.
2. Click the **Customize VCM** icon in the **Administration** section.
3. In the **Customize VCM** pane, select the merge type in the **All Merge Types** section.
4. In the **Available Merge Types** section, select each merge type and click **Add** or **Remove**.
5. Click **Save**.

Changing the View Compare/Merge Session Working Folder

View Compare/Merge allows you to check out files in a session to a temporary default working folder until you commit your merge changes. This allows you to test your changes to make sure they work before you commit them to the StarTeam repository on the server.

1. In your View Compare/Merge session, choose **VCM Session** > **Test Perspective** .
2. Choose **VCM Session** > **Working Folder** . The **VCM Session Working Folder** dialog box opens.
3. Select **Alternate** and type in, or browse to, the location for your working folder.

Changing View Compare/Merge Perspectives

When you start a View Compare/Merge session, the session opens in the **Merge Perspective**.

1. Open the view to use in your session.
2. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
3. Choose **VCM Session** > **Compare Perspective** , or **VCM Session** > **Test Perspective** to switch to one of those perspectives.
4. To switch back to the **Merge Perspective**, choose **VCM Session** > **Merge Perspective** .

Cloning a New View Compare/Merge Merge Type From One Server Configuration to Another

This procedure describes how to clone a newly created View Compare/Merge merge type from one server configuration to another.

1. Use the **Server Administration** tool to create the merge type completely for one server configuration, then click **Save**.
2. On the second server configuration, use the **Server Administration** tool to create the merge type by name only, based on the same View Compare/Merge default merge type, such as **Default.**

   Click **Save**.
3. Use the **Server Administration** tool to specify which projects use the new merge type and click **Save**.
4. From the StarFlow Extensions project on the first server configuration, check out `StarFlow Extensions/Projects/custom.vcm.xml`,

5. Copy the information between `<customVCMType name=...>` and `</customVCMType>` for this merge type and save it to the clipboard or another file.
6. From the StarFlow Extensions project on the second server configuration, check out `StarFlow Extensions/Projects/custom.vcm.xml`,
7. Replace the information between `<customVCMType name=...>` and `</customVCMType>` in this file with the information you saved to the clipboard or file.
8. Save the file and check it in.

Note: Cloning information from one `custom.vcm.xml` to another is slightly different. It assumes that you have a `custom.vcm.xml` in both servers and can paste the sections about merge types from one to the other. However, the XML files also contain information about the project and view structures for each server configuration and this information cannot be cloned. If you simply check the `custom.vcm.xml` from server configuration's StarFlow Extensions project into another server configuration's StarFlow Extensions project, you get error messages when you run View Compare/Merge.

Filtering Items in a View Compare/Merge Session

You can use filters in a View Compare/Merge session to limit the data shown on the upper pane, just as you do in a normal view in the StarTeam client. StarTeam provides a specific set of View Compare/Merge fields and predefined filters to use in View Compare/Merge sessions. You can also add new filters based on View Compare/Merge fields.

Note: Creating a new filter in a View Compare/Merge session is the same as in the main StarTeam client window, except that you need to ensure at least one View Compare/Merge field is included for a **Merge Perspective** filter.

Using a Predefined Filter

1. Click a component tab in the upper pane of the **Merge Perspective** or **Test Perspective**.
2. Select a filter from the **Filters** list.

   Each perspective contains a different list of filters. Also, there are two filter lists in the **Compare Perspective**: one for item types, and one for status and merge actions.

   Note: Only filters containing View Compare/Merge fields can be used in the **Merge Perspective**. View Compare/Merge fields are: **Default Action**, **Difference Type**, **Is Action Overridden?**, **Merge Action**, and **Merge Status**.

Saving the Current Arrangement as a Filter

1. Click a component tab in the upper pane of the **Merge Perspective** or **Test Perspective**.
2. Sort and group the data shown on the upper pane, as desired.
3. Right-click a column header in the upper pane and choose **Save Current Settings** from the context menu. The **Save Current Settings** dialog box opens.
4. Type a name for this filter in the **Filter Name** field.
5. Do one of the following:
   - Check **Public** to add this filter to the project so anyone with the appropriate access rights can use it.
   - Uncheck **Public** to make the filter private, available only to your user ID.
6. Click **OK**.

Running View Compare/Merge from the Command-line

The View Compare/Merge Utility (`VCMUtility`) is a command-line utility that compares a source StarTeam view to a target view, and optionally merges the differences into the target view.

1. Open a command window and go to the `StarTeam` installation root folder.
2. Type `VCMUtility`, followed by any desired options.

   `VCMUtility [<options file>] [options]`

   Tip: To see a list of all available commands and options, type `VCMUtility –help` in the command window.

Note: You can provide options in the specified `<options file>` (as the first parameter), command-line arguments, or both. Command-line arguments override any options found in the `<options file>`. In the `<options file>`, start option names in column 1 and exclude the leading "–".

Setting Locking Options for a View Compare/Merge Session

Locking source files prevents someone else from modifying them (in the tip configuration) while they involved in a View Compare/Merge session. Locks apply to the "whole item" within the view regardless of timestamp.

When starting a View Compare/Merge session, you can enforce default locking behavior by setting one or more locking options in the **View Compare/Merge Wizard**. Once in the session, you can override the lock status of any items.

When you close a View Compare/Merge session, you are prompted as to whether you want to retain the locks created by the View Compare/Merge session.

1. Open the view to use in your session.
2. Right click the item(s) and select **Advanced** > **View Compare/Merge**.
3. On the **Set Options** page of the **View Compare/Merge Wizard**, check the locking options you want to enforce in the View Compare/Merge session.

Tip: To view more lock-related fields in a View Compare/Merge session, right-click a column header and choose **Show** > **Fields** to open the **Show Fields** dialog box where you can select additional field columns to add to the upper pane.

Versioning Object Properties

The common properties of all objects that use versioning are summarized below.

External Name	Internal Name	Type	Description
Comment	`Comment`	text	This calculated property provides the object's revision comment regardless of whether it is stored in the `ShortComment` or `LCOMM` table.
CommentID	`CommentID`	int	If a revision comment < 2,000 characters exists, this field is a foreign key to the LCOMM table that contains the full comment.
Created By	`CreatedUserID`	int	ID of the user that first created the object.
Created Time	`CreatedTime`	timestamp	Date/time at which the object was created.
Deleted By	`DeletedUserID`	int	ID of the user that deleted the object. `-1` means the object is still "live".
Deleted Time	`DeletedTime`	timestamp	Date/time that the object was deleted. `0` means object is still "live". Non-zero means the record is soft-deleted.
Dot Notation	`DotNotation`	text	This calculated property is the dotted-notation representation of the object's version.

External Name	Internal Name	Type	Description
End Modified Time	`EndModifiedTime`	timestamp	Date/time that this object revision was replaced by another revision. `0` identifies the latest (tip) revision.
Locked By	`ExclusiveLocker`	int	ID of the user that currently has the object exclusively locked. `-1` means the object is not exclusively locked.
Modified By	`ModifiedUserID`	int	ID of the user that last modified the object.
Modified Time	`ModifiedTime`	timestamp	Date/time that the object was last modified.
My Lock	`MyLock`	int	This client-calculated property indicates if the current user has this object locked.
New Revision Comment	`NewRevisionComment`	text	This client-calculated property serves as a place holder for a new revision comment value.
Non-Exclusive Lockers	`NonExclusiveLockers`	text	List of users that currently have the object non-exclusively locked, if any.
Object ID	`ID`	int	ID that is unique to each object in the database.
Read Only	`ReadOnly`	int	This calculated property indicates if the object is currently read-only.
Revision Flags	`RevisionFlags`	int	This field is currently always zero.
Short Comment	`ShortComment`	text	Value of the revision comment if it is less than 2,000 characters.
Version	`RevisionNumber`	int	Revision number of this object revision. 0 = first revision.

Change Packages in View Compare/Merge

StarTeam has historically provided many features that supported change management (CM), including built-in workflow, customizable workflow, process links, process tasks, and View Compare/Merge. StarTeam also utilizes a comprehensive *change package* object, which allows you to track all changes made in a single commit. View Compare/Merge uses change packages instead of process tasks to record its changes.

Understanding Change Packages in View Compare/Merge

Change packages improve StarTeam’s ability to manage and track updates. A change package is an object that contains a set of changes applied to a target view. Change packages are created using the View Compare/Merge.

Note: Change packages in View Compare/Merge are not the same as workspace change packages associated with process items.

When you first start a View Compare/Merge session it acts as a staging area where changes are defined, reviewed, and tested. A View Compare/Merge session is visible in the target as a change package when the session is saved or committed. When the View Compare/Merge session is committed, its changes are atomically applied to the target view, and a change package object records the changes. Subsequently, a change package can be replayed, causing the same changes to be applied to another view.

Change packages represent sets of logically-related software artifact changes. A change package typically contains the changes needed to fix a bug or implement an enhancement, but it can include any set of updates defined by a user as a single logical change. A typical change package will contain mostly new file revisions, but it can also include related non-file changes such as marking a change request as fixed or updating a task as complete. A change package can include updates to any number of items of any type supported by StarTeam.

When showing changes for a selected item, the Cross-Platform Client shows both legacy and enhanced process links as well as links to relevant change packages. Links to change packages are more granular than links to process tasks. For example, links are created for update items other than files and folders, and links are created for updates not currently covered by process links such as deletes. The changes defined by a change package are applied as an atomic transaction which means that all of the changes are applied, or none of them are applied.

A change package also includes properties that represent metadata, such as a name, who created the change package, its purpose (description), who last modified the change package and when, and so forth. When a change package is committed, it is stored with the transaction ID with which its changes were applied. The transaction ID allows audit records and command traces associated with the change package to be identified.

Because change packages are persistent, you can browse change packages and their details. You can view all change packages applied to a view in chronological order, and for a given change package you can see both item- and non-item (metadata) change details. You can also query the change history of an item to determine which change packages, if any, were the source of any updates.

Note: Manual links cannot be created to change packages.

The Change Package Object

A change package is a StarTeam object that is persisted in the repository's database. A change package object is used to store information defined in a View Compare/Merge session when it is saved. In this state, a saved View Compare/Merge session is presented to users as owned by the target view. When the View Compare/Merge session is committed, it is still presented as belonging to the target view, but it is presented as a committed change package that can be queried and reused in additional ways.

A View Compare/Merge session can be created at the client and committed without ever being saved, consequently it is possible to create a change package object in the same transaction that applies its updates. When this occurs, the View Compare/Merge session is never shown as a saved change package. This is desirable for quick change sessions, for example, a View Compare/Merge promote of a few items.

The following are features of a change package object:

- Owned by a single view, the target view. It is on par with other view-owned objects such as labels and promotion states.
- Cannot be copied, moved, or shared to another view (although it can be replayed as described elsewhere.)
- Since a change package cannot be shared, the tip revision is always shown in a "current" configuration of the target view, and it cannot be pinned. Consequently, it does not have configuration properties such as **Branch On Change** or **Configuration Time**.
- Do not reside in folders. All change package objects belong to the view object and do not have path-related properties.
- A new revision is created each time an existing object is updated. A change package is updated whenever a View Compare/Merge session is saved or committed. Essentially, the pre-committed change package becomes the committed change package. A committed change package is read-only and can no longer be changed (except for its revision comment).
- Behave correctly with respect to "time travel". That is, when a user sets the target view configuration back in time, and change packages will appear as they did as of the rolled-back time.
- Can be attached to view labels. This allows, for example, users to attach a change package to a build label to indicate it was included as part of that build. Label attachment also allows two view labels to be compared to see their differences in terms of change packages. A change package also has a **Committed in Build** property which is set to `Next Build` when the change package is committed. The property is automatically modified to point to the next build label created. This feature allows change packages to identify the first build in which their changes were included.
- Prior to being committed, a change session can be deleted. As with all versioned objects, the delete is "soft" meaning the change package will no longer appear in the tip revision of the target view. However,

if the target view is rolled back in time, a previously deleted change package will “reappear” and can be opened in read-only mode. However, there is no way to “undelete” a change package. After a change package is committed, it cannot be directly deleted. Committed change packages are deleted only when the target view is deleted. Database information no longer needed by a deleted change package is removed during the database purge process.

- Can have queries and filters. This allows change packages to be displayed in the StarTeam Cross-Platform Client with popular sorting, grouping, and filtering criteria.

In addition to the StarTeam Server’s persistence and versioning service, change packages also use the server’s locking service. This means that a change package can be locked exclusively or with a shared lock. An exclusive lock is automatically applied to a saved change package when it is opened for editing. A change package can also be opened in read-only mode, though a non-exclusive lock is not applied for this use. Change packages cannot be “flagged” (bookmarked). Internally, change package objects also use attachments to store options and session state.

Change Package Security

Change packages have similar access rights as other view-level objects such as labels and promotion states. The ability to perform specific updates while committing a change session are derived from the permissions of the target view. For example, the ability to check-in a new revision of a file is derived from that permission in the target view.

An administrator can grant or deny the following view-level rights to any user or group:

Create change package Users who do not have this right for a given view can start a change session for that view (in the client application), but they can only save it to an external file format; they cannot save it persistently in the database.

Modify properties Users who do not have this right for a given view cannot create a new revision of a change package for that view, for example, by re-saving or committing an already-saved change package.

Delete from view Users who do not have this right for a given view cannot delete a saved change package in that view.

See change package and its properties Users who do not have this right for a given view cannot access change package objects for that view.

Change access rights Users who do not have this right can not modify change package access rights.

See history Users who do not have this right cannot see historic revisions of a change package.

Set exclusive locks Users who do not have this right cannot acquire and exclusive lock on a change package. this means that they cannot edit a saved change package.

Break exclusive locks Users who do not have this right cannot break an exclusive lock currently held by another user.

Label rights These rights control the ability to change what labels a change package is attached to. They include: **Attach/Adjust view labels**, **Detach view labels**, **Attach/Adjust revision labels**, and **Detach from revision labels**.

Note: Change package access rights are enforced with the same group/user/container resolution hierarchy as for other security ACLs.

Change Package Replay

You can select a committed change package and apply the same changes to another view. This capability is termed *replay*. A change package selected for replay is termed the *source change package*, and the view that it originally updated (originally the target view) is termed the *source view*. The new view to which the updates are to be replayed is termed the *replay target view*. The replay is identical in function to a View Compare/Merge merge in which the items updated by the source change package are selected as the source scope and merged to the replay target view.

When a change package is replayed to another view, a new change package is created for the new target view. The new session acts as the staging area for the replayed changes so they can be reviewed, adjusted, and so forth before being committed to the new target.

- If the replay target view is a parent of the source view, the replay is performed as a View Compare/Merge promote.
- If the replay target view is a child of the source view, the replay is performed as a View Compare/Merge rebase.
- If the replay target view is not an immediate child or parent of the source view, the replay is performed as a View Compare/Merge replicate.

View Compare/Merge Sessions

View Compare/Merge sessions become change packages when they are persisted at the server. When you initiate a View Compare/Merge session in the StarTeam Cross-Platform Client, or with the View Compare/Merge utility, it begins as an "in memory" session only. When the View Compare/Merge session is committed, a change package is created that stores the session's details.

View Compare/Merge sessions, and consequently change packages, can be created only for updateable view types, which include root (main) views, variant (branching) views, and non-derived (blank) views. A View Compare/Merge session cannot be created for a read-only reference view.

When you initiate a View Compare/Merge session, after the compare phase, the session is populated with changes proposed by View Compare/Merge after comparing items in a source view to items in the target view. The session is given a default change package name and description based on the source scope and merge type with which the session was initiated.

Before a View Compare/Merge session’s updates are committed, it can be saved to the StarTeam repository as a change package. The change package appears as a special type of object owned by the target view. A saved change package can be opened for further editing and saved any number of times. When its changes are committed, a change package becomes a committed change package and can no longer be edited.

Change Packages Can be Saved and Restored

When a change package is first created, it only exists as an “application session” in the StarTeam Cross-Platform Client or View Compare/Mergeutility. At any time before it is committed, it can be saved, causing its contents to be persisted in the StarTeam repository. Saving a View Compare/Merge session does not cause it to be “closed” in the StarTeam Cross-Platform Client. The user can continue working on the View Compare/Merge session and save it again at any time. Each save creates a new revision of the underlying change package object. The exclusive lock obtained in order to edit a change session is also retained when the session is saved. Only when the View Compare/Merge session is closed does the user have the choice to release the exclusive lock.

Change Packages Can be Opened and Edited

A saved change session appears as owned by the target view to which it applies. The StarTeam Cross-Platform Client provides a **Change Perspective** view to see saved change packages and open them. A change package must be opened in order to see details about items that have been compared or updated by the session.

By default, when a saved change session is opened, it is locked exclusively at the server. Consequently only one user can actively edit it at a time. If an exclusive lock cannot be obtained, the user can open the change package in read-only mode or attempt to break the exclusive lock.

Note: Breaking an exclusive lock requires the “break lock” privilege.

A user can always open a change package to which he has the read access right in read-only mode. A change package can be opened in read-only mode by multiple users. A change package opened from a rolled-back configuration of the target view can only be opened in read-only mode. Furthermore, the change package revision saved as of the rolled-back configuration is the revision opened. (This is useful, for example, to review information in an older revision of a change package.)

A change package can be opened, edited, and saved any number of times. When a change package is opened, it appears in a StarTeam Cross-Platform Client window as a resumed View Compare/Merge session.

When a change session is opened for editing, the only item-level changes allowed are changing View Compare/Merge action types (for instance, marking a difference as **ignore**) and resolving merge conflicts (for example, manually merging properties or file contents).

The only non-item edits allowed in a change package are:

- The change package’s Name and Description properties
- Pre- and post-commit revision and view label options
- The change package’s Revision Comment, which is used when the View Compare/Merge session is saved or committed
- The View Compare/Merge session’s Working Folder
- The View Compare/Merge session’s Default Comment, which is used as the revision comment for any new item revisions created during commit
- The View Compare/Merge sessions Responsibility, which is the person responsible for that View Compare/Merge session

Advantages of Change Packages

View Compare/Merge creates a change package object for tracking the changes in a View Compare/Merge change session. The change package is created in the target view and is not shared elsewhere. It is linked to all the items that were affected by the View Compare/Merge session.

Note: Historically, View Compare/Merge used to create a process task when committing a View Compare/Merge session, but now View Compare/Merge no longer creates a process task when it commits a View Compare/Merge session.

Other advantageous of change packages include:

- The comprehensive change package object allows you to track all changes made through View Compare/Merge sessions.
- Change packages can be saved before being committed so that others can review the proposed changes before they are committed. This can be done by sending a URL to a saved change package in an e-mail, for example.
- Every committed change package can be “replayed” to other views to which you want to apply the same set of changes.
- Perspectives in the StarTeam Cross-Platform Client window allow you to switch between the **Standard Perspective** and the **Change Perspective** by clicking icons which are right-justified in the toolbar.
- Using the **Change Perspective**, a manager, for example, can review all the change package objects and the details around those change packages, see what changes were committed to which views, and review changes in uncommitted change packages being proposed for committal.

Note: Once a change package has been committed, it becomes read-only and can no longer be changed.

Change Package Access Rights

Change Package item access rights available from the **Project** and **View** menu **Access Rights** menu item.

The following section describes generic change package rights.

Generic change package rights

Create change package
Allows the specified users and groups to create a change package using a View Compare/Merge session..

See change package and its properties
Allows the specified users and groups to view the change package and its properties.

Modify properties
Allows the specified users and groups to modify the change package properties for an uncommitted change package. From the **Change Perspective**, viewing the selected change package properties is read-only. However, if you select the change package and choose **Change Package** > **Open (in VCM)** , the change package opens in the VCM session where you can modify its properties.

The properties which can be changed are

Delete from view
Allows the specified users and groups to delete a change package from the view. This is an irreversible action.

Change access rights
Allows the specified users and groups to change the access rights for a change package. If you change this setting, be sure that you remain one of the users who can change access rights.

See history
Allows the specified users and groups to see change package history in the **History** pane.

Set exclusive locks
Allows the specified users and groups to lock change packages exclusively.

Break exclusive locks
Allows the specified users and groups to remove someone else’s exclusive lock on change packages.

Label rights

Attach/ Adjust view labels
Allows the specified users and groups to add a view label to a change package, and move a view label from one revision to another. This right controls direct manipulation of view labels for the change package at the item level. This right does not stop users from attaching a view label to the change package when a view label is created.

Detach view label
Allows the specified users and groups to remove a view label from the change package. Be aware that if users can delete view labels, they can detach a view label from the change package by deleting the view label from the view, regardless of the setting of this right.

Attach/ Adjust revision labels
Allows the specified users and groups to add a revision label to a change package and move a revision label from one revision to another. This right controls direct manipulation of revision labels for the change packages at the item level. It can stop users from committing change packages with an attached revision label.

Detach revision labels
Allows the specified users and groups to remove a revision label from a change package. Be aware that if users can delete revision labels, they can detach a revision label from the change package by deleting the revision label from the view, regardless of the setting of this right.

Link rights

See links Allows the specified users and groups to view the **Change Packages** in the **Change Package Perspective Replay Pane**.

Create links Allows the specified users and groups to commit a VCM Session.

Modify links Allows the specified users and groups to modify a link.

Delete links Allows the specified users and groups to delete a link.

Creating a View Compare/Merge Change Package

1. Open one of the two project views you want to compare in the StarTeam client.
2. Follow the steps in *Comparing and Merging Views in View Compare/Merge*.
3. Commit your change package session.
4. Save your change package session.

Note: If you use one of the **VCM** menu **VCM Session** menu items **VCM Session**, or **Export** after committing the session, StarTeam creates a uniquely named change package object in the target view for the change session which contains all the changes made in that View Compare/Merge change session. It then becomes available in the **Change Perspective**, where it can later be opened in read-only view, reviewed, have the properties modified, or have the change session Replayed which reuses the change package to apply the same set of changes to another view. Once a change package is committed, no more changes can be made to it. You can also use **VCM Session** > **Save**

Naming a Change Package

1. In the StarTeam Cross-Platform Client, open the target view of the change package.

   It can be the either the source view or the target view. Select either the entire view, or select specific items you want to compare or merge.
2. Right click the item(s) and select **Advanced** > **View Compare/Merge**. The **View Compare/Merge Wizard** opens.
3. You can choose to save the session before you resolve any conflicts or after you have finished and completed the session.
4. Click **VCM Session** > **Save** . The **Change Package Properties** dialog box opens where you can type in the name for your change package if you do not want to use the default name provided.

   Note: Once a change package has been saved, you can only change the name of an new, uncommitted change package. You can no longer change the name of a committed change package.
5. To change the name of a new uncommitted change package, select it in the **Change Perspective** to open it in the View Compare/Merge session. Choose **VCM Session** > **Properties** and change the name in the **Properties** dialog box.

Saving a View Compare/Merge Session to a Change Package

There are several ways to save a View Compare/Merge session: **VCM Session** > **Save** , and **VCM Session** > **Save to Local** . However, only **VCM Session** > **Save** saves the session as a change package.

Note: To open a session on a different computer, export the file instead of saving it by using **VCM Session** > **Export** menu. **Export** creates a View Compare/Merge exchange file (`.vcmx`) which allows the entire View Compare/Merge session to be transported to another computer, allowing the session to be imported from the **Project** > **Open Shortcut** menu on that computer.

1. Start a View Compare/Merge session and resolve the conflicts in it.
2. Choose **VCM Session** > **Save** . The **Change Package Properties** dialog box opens.
3. Make any desired changes to the change package properties and click **Save**.

   This saves the session as a change package object into the target view of the StarTeam Server.

Once you have saved your session, you can continue editing your session until you are ready to save it again or commit it.

Changing Access Rights for a Change Package

Change package access rights are view-level rights. They can be set at the project level as defaults for all views in the project, or for an individual view at the view level.

1. In the StarTeam Cross-Platform Client, open the project or view for which you want to set change package access rights.
2. Choose **Project** > **Access Rights** .
3. Click **Change Package** under the **Categories** list, then continue to specify the access rights for specific groups or people as in changing access rights for other objects.

Note: You can also change the Project or View access rights for a change package from the **Change Perspective**. To do that, click the **Change Perspective** icon, then perform the same steps for changing the access rights as stated above.

Deleting a Change Package

Prior to being committed, a change package object can be deleted. As with all versioned objects, the delete is “soft”, meaning the change package will no longer appear in the tip revision of the target view. However, if the target view is rolled back in time, a previously deleted change package will “reappear” and can be opened in read-only mode.

There is no way to “undelete” a change package. After a change package is committed, it cannot be directly deleted. Committed change packages are deleted only when the target view is deleted. Database information no longer needed by a deleted change package is removed during the database purge process.

Deleting a change package also causes it to be closed in the client. If any items were locked in the target view on behalf of the change package, the user is prompted as to whether to retain those locks or release them.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Select the change package.
4. Choose **Change Package** > **Delete** . This removes the `.vcms` and `.vcmx` session files that were previously saved locally or on the server.

Disabling a Change within a Change Package

You are able to disable a change within a committed change package. Disabling a change will cause the changed item to be ignored if the change package is replayed to another view. The **Status** column indicates the state of the changes within a change package: **Enabled** or **Disabled.**

1. Select a change within a change package.
2. Right click and select **Disable**. The **Status** column is updated with `Disabled`.

Editing a Change Package

You can only edit a change package with a status of **New** in the **Change Perspective**, which means it has been saved but not yet committed. This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

When a change package is opened for editing, the only item-level changes allowed are changing View Compare/Merge action types such as marking a difference as **ignore**, and resolving merge conflicts, for example, manually merging properties or file contents.

The only non-item edits allowed are: the change package's **Name** and **Description** properties, **Pre-and Post-Comment Revision** and **View Label** options, the change package's' **Revision Comment** which is used when the change package is saved or committed, the change package's **Working Folder**, and the change package's **Default Comment** which is used as the revision comment for any new item revisions created during commit.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Select the change package.
4. Click **Change Package** > **Open (in VCM)** to reopen the change package as an active View Compare/Merge session.
5. In the View Compare/Merge session, make the changes you want.
6. Save or commit the changed session when you are finished.

Change Package Properties

This topic contains the change package properties and their descriptions as displayed in the **Change Package Properties** dialog box.

Properties

The following properties are available for **Change Packages**. User's with the proper access rights can change these properties.

Name
Displays the textual identification of the change package. The name must be unique among all other change packages for the same target view. By default, StarTeam generates a name that contains a timestamp, which helps prevent duplicate names. This property is the change package object’s primary descriptor.

Description
Provides a full description of the change package. A typical use of the description property is to provide documentation not available in the change package’s process item such as notes to testers, reviewers, or developers.

Session Type
Defines the change package’s basic type. The type is set when the change package is first created and cannot be changed thereafter. It has the same values as VCM: `Rebase`, `Replicate`, and `Promote`.

Target View
Displays the name of the change package’s target view.

Source View
Displays the name of the change package’s source view.

Commit Time
Indicates the timestamp at which the change package was committed, or `null` if it has not yet been committed. This property cannot be directly modified by the user. Change packages that have been committed cannot be modified (except for the revision comment).

State
Indicates the current workflow state of the change package. This property will not be directly user-modifiable nor user-customizable. Furthermore, change packages will

not be integrated with user-customizable workflow, hence the values of this property are defined by actions performed to the change package as it evolves. The initial enumeration values of the state property are:

New This is the state that every change package receives when it is first created, but before any updates are defined for it.

Committed This state indicates that the change package has been committed, hence its changes have been applied. A committed change package and can no longer be modified (except for the revision comment).

Transaction ID Holds the ID of the transaction in which the change package's updates were applied. Prior to the `Committed` state, its value is `null`. Otherwise, the value can be used to query audit records in the **Audit** tab or commands used in the transaction by way of the StarTeam Server command trace file.

Pre-commit View Label Displays the name of the post-modification label that was created in the target view. This property is `null` if this label type was not created.

Pre-commit Revision Label Displays the name of the post-modification label that was created in the target view. This property is `null` if this label type was not created.

Post-commit View Label Displays the name of the post-modification label that was created in the target view. This property is `null` if this label type was not created.

Post-commit Revision Label Displays the name of the post-modification label that was created in the target view. This property is `null` if this label type was not created.

Committed by Displays the name of the user who committed the change package or `null` if it has not yet been committed.

Committed In Build Displays the name of the first build label created in the same view after the change package is committed of `null` if it has not yet been committed.

When a change package is committed, this property displays `Next Build` until a new build label is created.

Responsibility Displays the name of the user who is currently responsible for the change package. It is initially set to the user that first creates the change package.

If responsibility is changed to another user, and email notifications are enabled, the new user is notified of the change.

Other Server Services

In addition to the StarTeam Server's persistence and versioning service, change packages also use the StarTeam Server's locking service. This means that a change package can be locked exclusively or with a shared lock. An exclusive lock is automatically applied to a saved change session when it is opened for editing. A change session can also be opened in read-only mode, though a non-exclusive lock is not applied for this use.

Note: Change packages cannot be "flagged" (bookmarked).

Editing Change Package Properties

A selected change package's properties can be viewed in the **Change Perspective**, but when the change package is re-opened in the View Compare/Merge session, its properties can be edited with the **VCM Session Properties** dialog box. This dialog box servers several purposes. It allows you not only to view or change the properties of the change package, but it also allows you to commit the changes if the session was new, add or change the change session labels, and add a comment for the committed revision.

This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.

   Note: If the change package is committed it cannot be edited further.
3. Select the change package.
4. Click **Change Package** > **Open (in VCM)** to reopen the change package as an active View Compare/Merge session.
5. Choose **VCM Session** > **Properties** to open the **Change Session Properties** dialog box.
6. Modify the properties as desired.

   Note: If the session was uncommitted, you can change any of the properties for the session. However, if the session was committed before you reopened it in the VCM session, the only properties you will be able to change are the **Default** or **Alternate** working folder.
7. Click **OK**.

Linking Change Packages to Other Items

You can create manual links between change packages and other Items. This is especially useful when you want to update the process item of an already committed change package.

1. Right click a change package and select **Create Link**.
2. Select a folder or item for the end of the link.
3. Click the **Link** button on the toolbar to finish the link.

Opening a Change Package in a View Compare/Merge Session

This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Select the change package.
4. Do one of the following:

   If selected change package is new and uncommitted Click **Change Package** > **Open (in VCM)** to reopen the change package as an active View Compare/Merge session. Once a change package is opened in View Compare/Merge, you can view and modify it, and either save it again for more work later on, or commit it.

   If the selected change package is committed Choose **Change Package** > **Open as Read Only (in VCM)** .. Once the session is opened in View Compare/Merge, you can view it, but you can not make any changes.

   Note: Alternatively you can double-click a change package and it will open in a View Compare/Merge session. If it is committed, it will automatically open as read-only. The menu items for opening a change package in a View Compare/Merge session are also on the context menu of the selected change package.

Opening a Change Package in Read-Only Mode for Review

This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Select the change package.
4. Choose **Change Package** > **Open as Read Only (in VCM)** .

The selected change package opens in a View Compare/Merge session. You can view all the changes that were made, or view proposed changes to a yet uncommitted change package. However, you may not make any changes yourself or save the session.

Replaying a Change Package to Another View

The updates performed by a change package can be reapplied to another view. The **Change Perspective** in the StarTeam Cross-Platform Client lists committed change packages and provides a drag-and-drop gesture for applying a selected change package to another view. The new target view must be a root, branching, non-derived, or updateable reference view. Reapplying a change package to another view is called *replay*.

In a replay, the change package's original target view becomes the source view. Replaying a change package initiates a View Compare/Merge operation that selects the items updated by the change package as source items and merges them to the new target view. The replay uses a View Compare/Merge merge type that depends on the relationship between the source view and new target view:

- If the new target view is the immediate parent of the source view, the replay is performed as a View Compare/Merge promote operation.
- If the new target view is an immediate child of the source view, the replay is performed as a View Compare/Merge rebase operation
- If the new target view is neither the immediate parent or an immediate child of the source view, the replay is performed as a View Compare/Merge replicate operation.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. In the **Change Perspective**, depending on which layout you are using, click either the **Replay** icon on the right, or the **Replay** tab at the bottom.
4. Select the change package.
5. Drag that change package to the **Replay** viewer and drop it on the target view to which you want those changes applied.

   When a change package is replayed, a new change View Compare/Merge session opens in the new target view displaying the changes to be applied. The new session acts as a staging area for the replayed updates so that they can be reviewed, tested, and adjusted. Just as with all View Compare/Merge sessions, a session created by a replay can be saved, restored, and refreshed any number of times before it is committed.
6. Optionally, make any additional changes to the new change package, then either save the change package or commit the changes.

   When a View Compare/Merge session created by a replay is committed, a new change package is generated that belongs to the new target view. As a change package is replayed to other views, and as the resulting descendant change packages are replayed to yet more views, the replays form a "tree" that can be queried in the Cross-Platform Client showing the history of how a change package has been propagated to other views.

Note: Icons in front of change packages are different in the **Replay** viewer are different for new, uncommitted change packages than for committed change packages. The change package icon for a New change packages contains a red arrow. The committed change package icon is the same, but without the red arrow..

Using Filters and Queries in Change Packages

The StarTeam Cross-Platform Client allows users to list all active and committed change packages for a given view in the **Change Perspective**. The change package list can be sorted and filtered by various criteria. A change package can be selected, and certain details can be displayed. Only base properties such as the name, description, and lock state of uncommitted change packages can be viewed in the list. For committed change packages, details of each updated item can be viewed such as its properties and history. Other information about the change package, such as View Compare/Merge options used, can also be displayed. More details on interaction with change packages in the StarTeam Cross-Platform Client are described later.

This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Choose **Change Package** > **Filters** to open the standard StarTeam Cross-Platform Client **Filter** menu.
4. Create, modify, and select the various filters and queries you want to use for change packages in the **Change Perspective**. Use exactly the same processes you use to create filters and queries in the StarTeam Cross-Platform Client.
5. When you are finished, save any new or modified filters and queries. The new filters will display in the **Filter** list.

Viewing the Change Perspective

The StarTeam Cross-Platform Client has two perspectives available in the main window: The standard **Content Perspective** and the **Change Perspective**, the latter of which lets you view the details of all the changes made in committed View Compare/Merge sessions, or changes proposed in uncommitted View Compare/Merge change sessions. The **Change Perspective** also lets you apply the same changes made in one committed session to another project or view (*Replay*). This is possible because all the changes for a single View Compare/Merge session are recorded in a *change package* object created by the View Compare/Merge session when it is saved or committed. The **Content Perspective** also allows you to manage other aspects of a change package as you can with other StarTeam objects.

The StarTeam Cross-Platform Client automatically opens in the **Content Perspective**.

1. In the StarTeam Cross-Platform Client, click the **Show Change Perspective** icon at the right of the toolbar.
2. To switch back to the **Content Perspective**, click the **Show Content Perspective** icon at the right of the toolbar.

Viewing Change Package Details, History and Labels

This procedure assumes you have already saved one or more change packages that will be visible in **Change Perspective** of the target view in which it resides.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.
2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Choose one of the views you want to see from the icons at the right in the "side by side" layout, or from the tabs at the bottom of the "top over bottom" layout.

   Details pane Displays information specific to the selected change package, such as it's description, by whom it was created, when it was last modified, if it was committed, when it was committed and by whom. It also displays the session type (promote, rebase, or replicate), source and target view, transaction ID, and labels. The **Details** pane can be customized like the **Details** pane for other StarTeam items. Sample **Detail** pane templates are installed under the StarTeam Cross-Platform Client root installation folder in the `samples\details-templates` folder. There is a `changepackage.details.html` which you can use to modify the fields displayed on the **Details** pane in the **Change Perspective**.

   History pane Displays all the historical versions of the selected change package, complete with **Dot Notation**, **Name**, who modified each revision, and the **Description** of the revision. As with the **History** pane for other StarTeam items, there is a context menu for performing standard history actions, and right-clicking on the fields at the top let's you change which fields are displayed and the order in which they are displayed.

   Label pane This pane in the **Replay** view works exactly the same as for other StarTeam items. It shows all the branch revisions and their date/time information, and you have the same context menu as with other items to perform the same label functions.

Viewing Changes in a Change Package

This procedure presumes you have already saved and committed one or more change packages, which will be visible in **Change Perspective** of the target view in which it resides.

Note: Each change package is saved in the target view.

1. In the StarTeam Cross-Platform Client, open the target view of the change package.

   Note: Unless you have a view opened, there is no way to view the **Change Perspective**.

2. On the right side of the toolbar, click the **Show Change Perspective** icon to switch to the **Change Perspective**.
3. Select the change package.
4. Click the **Show Change Viewer** icon in the "side by side" layout, or click the **Change** tab in the "to over bottom" layout.

This displays the details of every change made in the change package, including among other things, each item type changed, the change type used, and the action used to resolve the conflict or difference.

Alternatively, you can open the change package as a Select the change package. session, but it will be in `Read-Only` mode since the changes have already been committed.

Note: You can also view the changes in a new (uncommitted session), but you must open a new change package as a Select the change package. session to see the proposed changes and resolutions.

Promotion States

A promotion state is a state through which a product passes. For example, most software products go through a release or production cycle – that is, the product moves from developers to testers and back again, until it is ready to go to the marketplace. Promotion states provide a convenient mechanism for ensuring that the right files or other items are available to the right people at the right stage of this cycle. For example, if a software administrator creates Test and Release promotion states, files that are ready for testing can be assigned to the Test state and files that have been tested successfully can be assigned to the Release state.

The promotion state feature permits an administrator to create promotion states and associate a view label with each state. An administrator creates a new promotion state configuration which is the basis for a new view or a reconfigured view containing only items with a specified promotion state. Administrators can also set access rights for promotion states. The view labels assigned to a promotion state are usually also used as build labels, so that they can serve as properties in change requests.

The view label for a state can be changed whenever appropriate. It can also be promoted from one state to the succeeding state. For example, although testers may always be using files in the Test promotion state, the files may be from Build 07 in one week and from Build 08 in the next. Users usually configure the project view for their job assignment by promotion state instead of by view label. For example, testers would configure their view to the Test promotion state.

Many features of the application depend on calculations involving times and dates. In particular, labels, configurations, and promotion states are all governed by time and date calculations. If the clients and the Server are not kept synchronized, a number of operations (such as checkouts, file status displays, or label creation) may fail or produce inaccurate or unreliable results.

Understanding Access Rights for Promotion States

Each view has its own set of promotion states. Access to these states is controlled by:

- The **Define Promotion Model** right which is set on the **View** node of the **Access Rights** dialog box for both projects and views. See “Granting Access Rights at the View Level”. A user with the **Define Promotion Model** right can do anything to the promotion model.
- Access rights that govern access to individual promotion states. These are **Generic Object Rights** and **Promotion State Specific Rights** which are set on the **Promotion State** node of the **Access Rights** dialog for both projects and views. They also appear on the access rights for individual promotion states.

The rights for an individual promotion state are checked at the state level; if necessary, the checking continues at the view level and eventually the project level. If a user is granted a given right at one level, there is no need to check the next.

- When a right is granted at the view level, it applies to all states in the view, unless access is denied at the state level.
- When a right is granted at the project level, it applies to all the states in all the views within the project, unless access is denied at the state or view levels.

Example of Using Promotion States

Suppose a software company wants to use the following promotion states to correspond with its use of the application:

Development Developers work with the tip revisions of files. These files have no view label because they are constantly changing. Many companies do not use Development as a promotion state, because configuring a view to a promotion state, even when the view label for the state is `<current>`, makes the view read-only.

White Box Test Testers check both the source code and the compiled executable file for problems that need to be fixed. The source code will have a view label to ensure that the testers are looking at an unchanging set of files. The view label will be attached to a promotion state named White Box Test. (White box testing is testing with full knowledge of what is in the source code.)

The executable files are not stored in the application because they can be easily built from the source code. Testers install them from a `Builds` folder on the network. This folder has child folders named `Build 1`, `Build 2`, and so on.

Change requests are entered against the executable files only. Developers make repairs in the current source code, sometimes reviewing the files with the view label attached to the Black Box Test promotion state.

Black Box Test

Testers install the executable file, just as they would with white box testing. However, they do not need to see the source code or use promotion states with it. (Black box testing is testing with no knowledge of what is in the source code.)

Change requests are entered against the executable files only. Developers make repairs in the current source code, sometimes reviewing the files with the view label attached to the Black Box Test promotion state.

Alpha Test

End users of the software product being developed install the product executable files and test the product in their own environments.

Change requests are entered by the alpha coordinator and/or the users against the executable files only. Developers make repairs in the current source code, sometimes reviewing the files with the view label attached to the Alpha promotion state.

Beta Test

Beta testing is similar to alpha testing, but the group of users is greatly expanded because the product is much more stable.

Change requests are entered by the beta coordinator and/or the users against the executable file only. Developers make repairs in the current source code, sometimes reviewing the files with the view label attached to the Beta promotion state.

Release

The product is now sold in the marketplace. Users install the executable file and call product support. Product support enters change requests against the executable files only. Developers make repairs in the current source code, sometimes reviewing the files with the view label attached to the Release promotion state.

The fixes go into future product releases and service packs to releases already in the marketplace.

In this example, every time the source files are used to produce a build (set of executable files) for testing, a view label is applied to the files to identify them for future reference. It is convenient to use view labels such as **Build 1**, **Build 2**, and so on, so that it is clear which source code files were used to create which set of executable files.

Over time, the build or view label associated with a promotion state will change. For example, the Release state may initially be associated with `<current>`, rather than a view label, because no files are candidates for release and no appropriate view label has been created. When white box testers decide that the set of files that they have examined is ready for black box testing, the view label associated with the White Box Test promotion state will be moved to the Black Box Test promotion state, and so on.

If promotion states are used, developers and testers who look at source code do not need to know that view label **Build 120** is currently being checked by white box testers, that the executable files for **Build 117** are currently undergoing black box testing, and other details.

Configuring Promotion States

When creating promotion states, many administrators assign `<current>` to the initial promotion state instead of a view label, because that state always uses the tip revisions. They also often assign `<current>` to later promotion states for which no view labels currently exist. These states may receive a view label later, when the files associated with a view label meet the criteria required by the state. Alternatively, a view label may be promoted from the preceding state to the label-less state.

Creating a New Promotion State

Note: You can create promotion states only if you have the required access rights, which are found at the project or view level.

1. Click **View** > **Promotion**. The **Promotion** dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Click **Add** to open the **Promotion State** dialog box.
3. Type the **Name** and **Description** of the promotion state.
4. Assign a view label to this state by selecting it from the **View Label** drop-down list. The labels are listed in reverse chronological order, based on the time at which they were created. You can change the label when appropriate by using this dialog box or assign it to the next state by promoting it.
5. Click **OK** to close the **Promotion State** dialog box, and **OK** again to close the **Promotion** dialog box.

Editing or Deleting a Promotion State

1. Click **View** > **Promotion**. The Promotion dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Select a promotion state.
3. Click **Edit**. The **Promotion State** dialog box opens.
4. Modify the **Name**, **Description**, or **View Label**.
5. Click **OK** to close the **Promotion State** dialog box, and **OK** again to close the **Promotion** dialog box.

Moving the Promotion State Up or Down

1. Click **View** > **Promotion**. The **Promotion** dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Select a promotion state from the list and click **Move Up** or **Move Down**.

Modifying Access Rights of a Promotion State

1. Click **View** > **Promotion**. The **Promotion** dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Select a promotion state from the list and click **Access Rights**. The **Promotion State Access Rights** dialog box displays.
3. Do one of the following:
   - Click **Add** to specify a group or specific user to grant or deny specific access privileges. Check the specific access rights and select **Grant** or **Deny**.
   - Select an existing user or group and change which access privileges they have.

Promoting View Labels

You can promote a view label from one promotion state to the next if you have the appropriate access rights.

1. Click **View** > **Promotion**. The **Promotion** dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Select the promotion state currently associated with the view label that you want to promote.
3. Click **Promote**. The **Promote View Label** dialog box indicates that the view label is now associated with the next state (the state immediately above the selected state in the **Promotion** dialog box).
4. Verify that the information is what you were expecting to see.

5. Click **OK**. The selected view label now applies to two promotion states: the one to which it was promoted and the one you originally selected. Usually, your next action is to associate a new view label with the original state.

Promotion State Access Rights

Each view has its own set of promotion states. Access to these states is controlled by the “Define promotion model” right, which is available from the **View** node of the **Access Rights** dialog at the view and project levels. A user with the `Define promotion level` right can do anything to the promotion model, for example create and delete states, edit their properties, promote a label from one state to another. Promotion is a subset of editing properties. Anyone who can edit the properties of a state can also promote that state and reorder the states within the view.

Access rights that govern access to individual promotion states. These **Generic object rights** and **Promotion state specific rights** are available from the **Promotion State** node of the **Access Rights** dialog at the view and project levels. They also appear on the access rights for individual promotion states. The rights for an individual promotion state are checked at the state level; if necessary, the checking continues at the view level and eventually the project level. If a user is granted a given right at one level, there is no need to check the next.

When a right is granted at the view level, it applies to all states in the view, unless access is denied at the state level. When a right is granted at the project level, it applies to all the states in all the views within the project, unless access is denied at the state or view levels.

Generic object rights

Change object access rights	Change the access rights for an individual promotion state. If you change this setting, be sure that you remain one of the users who can change access rights. This right is a generic object right. After creating a promotion state, you must exit and reenter the Promotion dialog if you want to set access rights for the newly created state.

Promotion State specific rights

Modify label assignment	Change the label assigned to an individual state either by clicking the Promote button or editing the label property. No other properties for the state can be edited unless the user also has the Define promotion model access right from the View node. This right is a promotion state specific right.

Setting Promotion State Access Rights at the Project or View Level

Setting promotion state access rights is very similar to setting other access rights. The access rights can be set at the project or view level as well as for individual promotion states.

1. Select **Project** > **Access Rights** or **View** > **Access Rights** . The **Access Rights** dialog box opens.
2. Select the **View** node.
3. Click **Add**. The **Assign Access Rights To** dialog box opens.
4. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
5. Select **Grant** and click **OK**.

   ⚠ **Caution:** Never select **Deny** unless you are creating an exception. Deny records must be created before grant records.
6. Select **Grant** and click **OK**.
7. Select/clear the appropriate check boxes. Selecting or clearing the check box for a category, such as **Generic object rights** for a project, selects or clears all the access right check boxes for that category.

The category check box has only two states. When it is cleared, the access right check boxes for that category are either all cleared or mixed: some selected and some cleared.

Caution: Clicking **Delete** removes the selected user or group from the **User and Groups** list in the **Access Rights** dialog box. The selected user or group loses any previously set access rights to the Server.

8. Click **OK**.

Setting Access Rights for Individual Promotion States

1. Select **View** > **Promotion** . The **Promotion** dialog box opens.
2. Select a promotion state from the list.

   Note: You can create the promotion state in the **Promotion** dialog box. However, you must click **Apply** before you can set access rights. After you click **Apply** (or close and reopen the dialog), the **Access Rights** button is enabled.
3. Click **Access Rights**. The **Promotion State Access Rights** dialog box opens.
4. Click **Add**. The **Assign Access Rights To** dialog box opens.
5. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
6. Click **OK**.

Labels

In version control, the term `label` corresponds to the act of attaching a view or revision label (name) to one or more folders/items. StarTeam enables you to create two types of labels:

View labels Are automatically and immediately attached to all folders and items in a view at the time you create the view label. View labels have multiple purposes, but you primarily use them to place a *time stamp* on the entire contents of the view and as *build labels*. When you roll back the view to that label, you see everything that existed at that point in time—unless the label has been adjusted. You can create a view label for a specific point in time or as a copy of another existing view label. Unless the view label is frozen, you can adjust it to include or exclude some folders and items by attaching or detaching view labels. You can also move a view label from one revision to another.

Revision labels Are not attached automatically to any item in the view. Instead, they are used to designate a set of folders or items within a view. For example, you might want to label a group of files that should be checked in and out together. After you create a revision label, you attach specific items, building it up to reflect a specific set that is typically a small subset of the view. StarTeam can automatically attach new file revisions to a revision label at check-in time if you like.

About Labels

You can attach a label to any type of StarTeam item, including folders, files, requirements, change requests, tasks, topics, and audit entries. Any item can have more than one label. However, no two revisions of the same item can have the same label at the same time.

Every label is unique within its view. That is, no view label can have the same name as any other view label, no revision label can have the same name as any other revision label, and no view label and revision label can have the same name. Each view has its own set of labels. This also means that every label has a *name* that must be unique from other labels belonging to the same view. Each label also has a *description* that helps users understand the purpose of the label.

You can manually attach or detach both view labels and revision labels to or from a folder or item. In addition, you can use either type of label to identify a file when it is checked out. When you check a file in, you can attach and create a revision label for that file or attach an existing revision label.

You can select any type of item by its label. For example, you can select all files with a particular revision label and roll them back to that label, making the revision with that label the tip revision. Then you can compare your working files to the rolled-back revisions.

You can set access rights for labels at the view level or at the folder or item level. You must grant the rights to create labels, edit their properties, and delete them at the view level. However, you can grant the right to move a label (also called *adjust a label*) at the folder or item level.

A label can be frozen, which means no new artifacts can be attached to it, and attached artifacts cannot be detached nor reattached at a different revision. Conversely, non-frozen labels can have all of these modifications. Since many organizations depend on the immutability of frozen labels, a specific security permission is required to freeze or unfreeze a label.

StarTeam actually attaches specific *item revisions* to a label, each of which infers a specific artifact revision. Since each item specifies a parent folder, the artifact is attached in a specific folder. This means that if you move an item, you need to reattach it to any label for which you want to reflect the artifact in its new folder. If you don't reattach a moved item to an existing label, it will continue to be attached to the label in its old folder (which may be what you want).

Even if frozen, both view and revision labels can be cloned. That is, you can create a new label starting out identical to an existing label and then adjust the revisions attached to it. A common practice is to clone a previous label, attach only new file revisions that were created to fix a bug, and use the new label to identify the file revisions for a new build candidate or release candidate.

Time Stamps and Build Labels

Using a view label as a *time stamp*, you can roll a view back to see everything in the view as it was at the time the label was attached. For example, to see if a particular file was in the beta version of a product, you can roll back the view to the beta label.

You may also use a view label as a *build label*, which allows the QA team to immediately determine what build to test for a fix to any given change request. To use a view label for this purpose:

- It must be designated as a build label.
- It must be created while the **Addressed in build** property for the change request contains the value `Next Build`.

When StarTeam creates the label, each change request with Next Build as its **Addressed in build** property will be reset to the build label.

To create a view label, you must select the current configuration of the view. Historical configurations are read only, and adding a label is considered a change. However, if a label already exists for a prior configuration, you can adjust its name, files and folders can be added to it or detached from it, and so on. You can also move a view label from one revision to another.

For example, suppose your administrator creates a view label before each build, giving that label to the tip revision of each file in the view, and then checks out all the files with that label for the build. If the tip revision of one file does not change for a few weeks (or longer), it can acquire several view labels, while a file that changes frequently may have several revisions with no view labels and other revisions with only one view label.

When you detach a view label from a folder, StarTeam automatically detaches the label from everything in the subtree for which the folder is the root. If you roll back a view to a specific view label and a folder does not have that label, you cannot see the children of that folder and their contents anyway.

You can only create a view label at the view level and only while the configuration is current. However, you can create a view label for the current configuration or for a time in the past. In either of these two cases, StarTeam attaches the new label to the tip revision of each folder, file, change request, task, or topic that belonged to the view at the specified time.

You can also create a view label as a copy of an existing view label or as a copy of the view label currently attached to a promotional state. In these two cases, StarTeam attaches the new label to exactly the same items and revisions as the existing view label.

You can check file revisions out using this label or roll back the view to this label and see all the items associated with that label. For example, if you create the view label *Build 100* as you make Build 100 of your product from a view, all the files in the view will have the label *Build 100*.

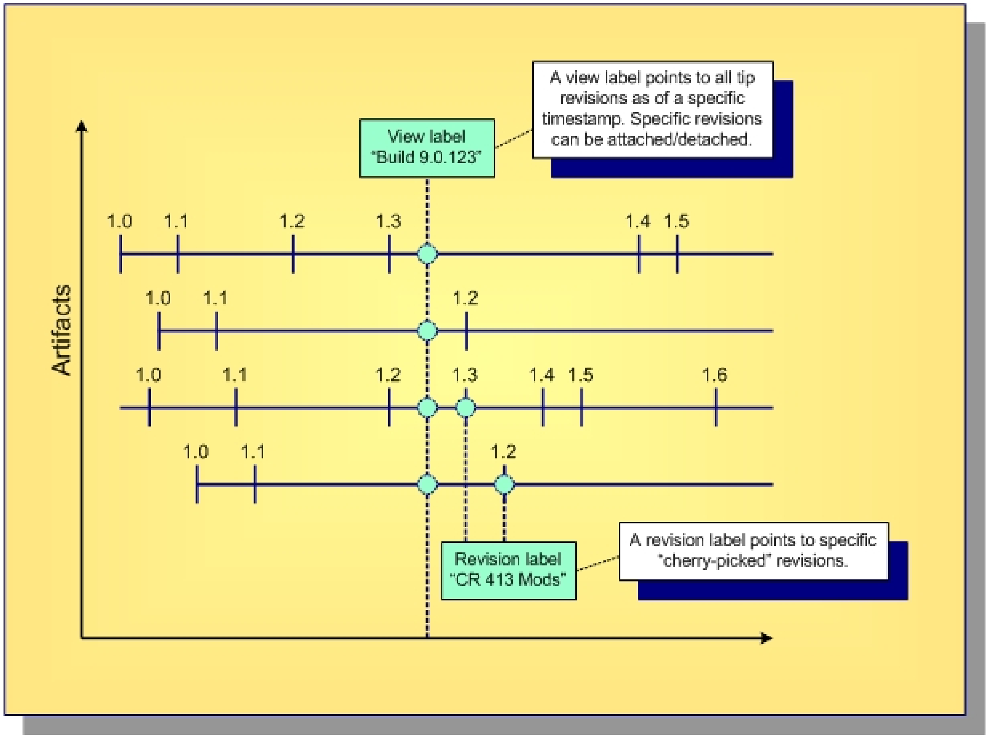
If some items should not be included, you can detach the label from those items individually. For example, if some files should not have that label, select the files then select **Labels** > **Detach** from the **File** menu or context menu to detach that label. If the files that should not be included all belong to the same folder and are the only files in that folder, use the **Labels** command on the **Folder** menu. For example, if the help files were not checked in until after the view label was attached, you can move that label from the previous revisions of the help files to the newly checked-in help files.

Label Access Rights

You can set access rights that apply to labels at the view level and at the folder/item levels. You set the access rights that allow a user or group to create labels, edit their properties, and delete them at the view level. For example, if you can create a label, you can set its initial properties. However, if you do not have the right to edit label properties, you cannot later freeze or unfreeze that label.

You can attach labels to individual folders or items, detach from them, or move from one of their revisions to another. The access right to move a label is named **Adjust a label**. You can grant or deny these rights at the folder or item level.

Example View and Revision Labels


A view label points to all tip revisions as of a specific timestamp. Specific revisions can be attached/detached.
View label "Build 9.0.123"
1.0
1.1
1.2
1.3
1.4
1.5
1.0
1.1
1.2
Artifacts
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.0
1.1
1.2
Revision label "CR 413 Mods"
A revision label points to specific "cherry-picked" revisions.

Note: Not all items in a view have to be attached to a label. Conversely, an item can be attached to any number of view and revision labels as you like, but only one revision of an item can be attached to any specific label.

View Labels

View labels should be used to mark configurations of the entire view that match specific milestones such as a new build number, a point from which another view was created, a release candidate, and so forth.

Consequently, you don't create or use view labels for specific check-ins. Instead, you create them when one or more changes have been committed and it's time to launch a new build, spawn a release view, or promote changes to another view. The following two scenarios illustrate good uses of view labels.

The following two scenarios illustrate good uses of view labels.

Scenario 1: Daily Builds

Whether your team prefers daily builds, nightly builds, hourly builds, or builds on demand, this approach creates new view labels where "Use as build label" is selected:

- To get the process going, first create a view label as of a specific timestamp and launch the build process. (Some users prefer to have the build process itself create the view label.)
- The build process then checks out files attached to that label and launches compiles, links, and other build tasks. If these tasks are successful, the same build process could also launch unit tests and other verification tasks.
- If all build tasks are successful, the view label could be marked as "frozen", which identifies it as a "good build".
- Conversely, if a build task fails, the build process could generate a report or notify someone via email. If you're early in the development activity, you might choose to just move forward, allowing the team to continue making changes in the view. The next build process will simply "try it again". When you're later in the development activity, you'll want to have the appropriate developers fix their errors, re-attach the new revisions they create to the same view label, and perform the build again. Only when you achieve a "good build" is the view label marked frozen.

The advantage of this approach is that it tends to ensure that the tip revisions in a view are generally buildable. This supports a growing software development practice known as *continuous integration*. The disadvantage of this approach is it may be difficult with large teams and environments with lots of changes. It can result in a lot of broken builds, finger pointing, and "nasty gram" emails.

Scenario 2: "Change" Builds

Instead of relying on tip revisions being generally stable and buildable, another approach is to create view labels that are attached to revisions that are carefully selected. The steps that take this path are outlined below:

- Assume you start with an existing "good build" view label. As with the previous scenario, this label would be flagged as a build label and probably frozen.
- Although many changes are occurring in the build, you want to select only specific changes as candidates for inclusion in the next "good build" label. To do this, ensure that the corresponding file revisions are attached to a revision label and that this label is **only** attached to the file revisions you're interested in.
- Start the next "good build" label by "cloning" the current label. In the StarTeam Cross-Platform Client, select **View** > **Labels** > **View tab** > **New**. In the corresponding dialog, choose **Labeled configuration** and select the current `good build` label. Your new label will now be identical to the old label.
- Select the file revisions associated with the desired changes. In the **File** tab, select **Select** > **By Label** and choose the appropriate revision label.
- Now attach these revisions to the cloned view label. In the **File** tab, select **Labels** > **Attach**. In the corresponding dialog, select the cloned label in the upper label list. In the "Attach to items at" group box, select "Labeled configuration" and choose the same revision label you used in the previous step. This ensures that the correct revision of each file is attached to the cloned label, otherwise the tip revision will be attached, which may be the wrong revision.
- Repeat the previous two steps if there are multiple revision labels representing file revisions that should be included in the new label.
- Now launch the appropriate build and test process for your new "good build" label. Mark the label frozen or reattach fix revisions and retry as in the previous scenario depending on whether the build/test process succeeds.

This approach allows you to tag view configurations as candidates for builds, promotes, etc. without relying on the tip revisions being stable. The disadvantage of this approach is that if your latest “good build” label is way behind the view’s tip configuration, the quality of more recent changes may not be known for a while (which goes against the premise of continuous integration.)

Because this approach employs label “cloning”, there is a caveat we should mention with respect to deleted items. Suppose an item is attached to a view label and then deleted from the view. As you’d expect, when you “time travel” by adjusting your view window back to the view label, the item “reappears” because it existed when it was attached to the label. Less obvious, however, is that when you clone the label, the item will also be attached to the new label because the new label is initially identically to the old label. If you don’t want items deleted in the tip configuration to be attached to a cloned label, just detach them from the cloned label.

Creating Revision Labels

Like view labels, new revision labels can be created from the **View** menu. In fact, if you are creating a new revision label based on an existing revision label in another view, you must use the **View** menu for this purpose. However, revision labels can also be created from the **Folder Tree** menu, a component menu, or the context menu.

1. Select a folder from the folder tree.
2. Select one or more items in the upper pane.
3. Choose **View** > **Labels** . The **Labels** dialog box opens.
4. Click the **Revision** tab. The labels are listed in reverse chronological order based on the time at which they were created.
5. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
6. Optionally, check **Frozen** to freeze the label so that revisions attached to it cannot be changed.
7. Click **OK**.

Creating View Labels

View labels, usually used as build labels by default, can be extremely useful when you want to label every folder and item in a particular view.

1. Open the view to which you want to apply the label.
2. Choose **View** > **Labels**. The **Labels** dialog box opens.
3. Click **New**. The **View Label** dialog box opens.
4. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
5. Select one of the following:

   Current Configuration Attaches the label to the tip revision.

   Labeled Configuration Attaches the label to the revision with a specified label. The labels are in reverse chronological order based on the time at which they were created.

   Promotion State Configuration Attaches the label to the revision currently in a specified promotion state. (Actually, the label is attached to the revision that has the promotion state's current view label.)

   Configuration As Of Attach the label to the revision that was the tip revision at a specified date and time.

6. Optionally, check **Use As Build Label** to update each change request that has **Next Build** as the setting for its **Addressed In Build** property. If this option is not selected, the view label will still be attached to change requests, but the setting of the **Addressed In Build** property will not change.

7. Optionally, to freeze the label so that the revisions attached to it cannot be changed, check **Freeze**.
8. Click **OK**.

Note: It is always important to synchronize the dates and times of the computers that run the StarTeam clients and the StarTeam Server. However, if they are not synchronized and you select the current time as a label's configuration, the label may not be immediately visible.

Configuring or Viewing Label Properties

View label properties include a name, description, frozen/unfrozen status, configuration, and build label status. Revision label properties include a name, description, and frozen/unfrozen status.

Displaying View or Revision Label Properties for Editing

1. Select a folder from the folder tree.
2. Choose **View** > **Labels** . The **Labels** dialog box opens.
3. Select a label on the **View** or **Revision** tabs and click **Properties**. The **Edit Label** dialog box opens enabling you to modify the label name or description, and freeze or unfreeze the label.

Displaying Folder Label Properties

1. Select a folder from the folder tree.
2. Right-click the selected folder and choose **Labels**. The **Labels** dialog box opens showing all the labels attached to the folder.
3. Select a label from the list and click **Properties**. A read-only **View Properties** dialog box opens enabling you to view the properties for the selected label.

Displaying Label Properties from the Label Pane

1. Select an item in the upper pane.
2. Click the **Label** tab in the lower pane.
3. Right-click a label in the **Label** pane and choose **Properties**. The read-only **View Properties** dialog box opens enabling you to view the properties for the selected label.

Attaching Labels to Folders

Labeling folders is slightly different from labeling items. When you attach a revision label to a folder, you can also attach it to the items that the folder contains and to everything in the subtree for which the folder is the root (its child folders and their contents).

If you detach a revision label from a folder, you can also detach the label from the items associated with the folder and, optionally, from the child folders and their items. If you detach a view label, the label is automatically detached from the items that the folder contains, from the child folders, and from their contents.

Note: To determine whether a label is a revision label or a view label, double-click the label (or select the label, and then click **Properties**). A revision label has a name and a description. A view label has a name, description, and a configuration time.

Creating a New Revision Label and Attaching it to a Folder and its Contents

1. Select a folder from the folder tree.

2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
3. Select the revision that will receive the new label.
4. Right-click the selected item(s) and choose **Labels** > **New** . The **Attach a New Revision** dialog box opens.
5. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
6. Optionally, check **Frozen** (that is, cannot be changed) to ensure that only the selected revision can have this label.
7. Select one of the following:

Folder Only	Attaches a label to the selected folder.
Folder and Items Contained in Folder	Attaches a label to the folder and its items.
Everything in Subtree Rooted at Folder	Attaches a label to the folder, its items, and its child folders and their items.

Note: Because attaching a label to a folder also allows it to be attached to the folder contents, children, and so on, the label is always attached to the current configuration of each folder and item. You cannot label a prior revision of a folder.

Attaching an Existing View or Revision Label to a Folder and its Contents

1. Select a folder from the folder tree.
2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
3. Click **Attach**. The **Attach a Label** dialog box lists all the existing labels and identifies them as view or revision labels. By default, both the **View Labels** and **Revision Labels** check boxes are selected.
4. To display only view labels or revision labels, uncheck the appropriate check box.
5. Select a label.
6. Select one of the following:

Folder Only	Attaches a label to the selected folder.
Folder and Items Contained in Folder	Attaches a label to the folder and its items.
Everything in Subtree Rooted at Folder	Attaches a label to the folder, its items, and its child folders and their items.

7. Click **OK**.

Note: Attaching a label to a folder always attaches it to the current configuration of each folder and item. It is not possible to label a past revision of a folder, although you can do so for items.

Reviewing the Labels Attached to a Folder's Revisions

1. Select a folder from the folder tree.
2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
3. The **Labels** dialog box lists all labels currently attached to this folder on a revision-by-revision basis.

Moving a Revision Label from one Folder Revision to Another

1. Select a folder from the folder tree.
2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.

3. Drag a revision label from one folder revision node to another in the **Labels** dialog box.

Attaching Labels to Items

If you are dealing with an item or set of items that you want to group together, you can create a new revision label, attach an existing label to the item or an item revision, review all labels, or move a revision label.

Note: A Label can be attached to only one revision of an item.

Creating a New Revision Label for Selected Items

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Select one or more items in the upper pane.
4. Right-click the selected item(s) and choose **Labels** > **New** . The **Attach a New Revision** dialog box opens.
5. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
6. Optionally, check **Frozen** to ensure that only the selected item revision can have this label.
7. Indicate what item revision is to receive this label by selecting a configuration option. The choices are:

   Current Configuration Attaches the label to the tip revision.

   Labeled Configuration Attaches the label to the revision with a specified label. The labels are in reverse chronological order based on the time at which they were created.

   Promotion State Configuration Attaches the label to the revision currently in a specified promotion state. (Actually, the label is attached to the revision that has the promotion state's current view label.)

   Configuration As Of Attach the label to the revision that was the tip revision at a specified date and time.

8. Click **OK**.

Attaching an Existing View or Revision Label to Selected Items

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Right-click the selected item(s) and choose **Labels** > **Attach** .

   The **Attach a Label** dialog box opens. This dialog box lists all existing labels and identifies them as view or revision labels. By default, both the **View Labels** and **Revision Labels** options are checked.
4. Select a label from the list.
5. Indicate what item revision is to receive this label by selecting a configuration option. The choices are:

   Current Configuration Attaches the label to the tip revision.

   Labeled Configuration Attaches the label to the revision with a specified label. The labels are in reverse chronological order based on the time at which they were created.

   Promotion State Configuration Attaches the label to the revision currently in a specified promotion state. (Actually, the label is attached to the revision that has the promotion state's current view label.)

Configuration As Of	Attach the label to the revision that was the tip revision at a specified date and time.

6. Click **OK**.

Attaching an Existing View or Revision Label to a Specific Item Revision

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Right-click an item revision in the **Label** pane and choose **Attach**.

   The **Attach a Label** dialog box opens. This dialog box lists all existing labels and identifies them as view or revision labels. By default, both the **View Labels** and **Revision Labels** options are checked.
4. Uncheck **View Labels** or **Revision Labels** to limit the list to one specific type of label.
5. Select a label from the list.
6. Indicate what item revision is to receive this label by selecting a configuration option. The choices are:

Current Configuration	Attaches the label to the tip revision.
Labeled Configuration	Attaches the label to the revision with a specified label. The labels are in reverse chronological order based on the time at which they were created.
Promotion State Configuration	Attaches the label to the revision currently in a specified promotion state. (Actually, the label is attached to the revision that has the promotion state's current view label.)
Configuration As Of	Attach the label to the revision that was the tip revision at a specified date and time.

7. Click **OK**.

Reviewing All Labels Attached to Item Revisions

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Select the item in the upper pane, then click the **Label** tab in the lower pane. This displays the **Label** pane which shows all revisions for the item.
4. Right-click a revision in the **Label** pane to display all of its labels.

Moving a Revision Label from One Item Revision to Another

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Select the item in the upper pane, then click the **Label** tab in the lower pane. This displays the **Label** pane which shows all revisions for the item.
4. Right-click a revision in the **Label** pane to display all of its labels.
5. Select a specific label and drag it from one revision to the another.

Demoting View Labels

Sometimes a labeled set of files is promoted prematurely and must be demoted. For example, if a specific build is promoted to the `Beta` state, but contains serious flaws, it should probably be returned to the prior promotion state. You can only demote view labels by editing the promotion state.

1. Click **View** > **Promotion**.

2. Click **Edit**. The **Promotion State** dialog box opens.
3. Select a different view label from the **View Label** list.
4. Click **OK**.

Promoting View Labels

You can promote a view label from one promotion state to the next if you have the appropriate access rights.

1. Click **View** > **Promotion**. The **Promotion** dialog box displays the states currently created for this view. The final state appears at the top of the list.
2. Select the promotion state currently associated with the view label that you want to promote.
3. Click **Promote**. The **Promote View Label** dialog box indicates that the view label is now associated with the next state (the state immediately above the selected state in the **Promotion** dialog box).
4. Verify that the information is what you were expecting to see.
5. Click **OK**. The selected view label now applies to two promotion states: the one to which it was promoted and the one you originally selected. Usually, your next action is to associate a new view label with the original state.

Copying Revision Labels

Occasionally, you may want to copy a revision label. For example, if you move or share an item from one view (source view) to another (target view), labels from the source view do not become part of target view. However, by copying the revision labels after the move or share, you can selectively maintain revision labels on the moved or shared items.

Copying a revision label immediately attaches it to the same revisions of the same items as the original revision label. If the two revision labels are in the same view, each label will be attached to the same number of items. However, if the two revision labels are in different views, the new label becomes attached to the same revisions of the same items only if the items and their revisions exist in the new label's view at the time of the copy operation.

Although you can copy revision labels in a variety of ways, the following procedure allows you to copy a revision label whether it is in the current view or in another accessible view. It assumes that you are dealing with files, but can be adapted for other types of items.

1. Choose **View** > **Labels** . The **Labels** dialog box opens.
2. Click the **Revision** tab.
3. Click **New**. The **Revision Label** dialog box opens.
4. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
5. Check **Copy From Another Revision Label**.
6. Click **Select** to open the **Copy a Revision** dialog box.
7. Select a project from **Project** from the list, a view in the **View** tree, and a revision label in the **Labels** list.
8. Click **OK**.
9. Click **OK**.
10. Click **Close**.

    The new revision label is attached to the same revisions as the existing label.
11. Do one of the following:

    - Check in the changed file or files using the new revision label.

- Check in the changed file and attach the new revision label manually to the changed file revisions. To do this, select the checked in file and click the **Label** tab in the lower pane. Drag the new revision label to the correct (probably tip) revision in the **Label** pane. Repeat for any other changed files.

  Note: If you have added a new file, use **File** > **Labels** > **Attach** to attach the label.

Copying View Labels

Occasionally, you may want to create a view label and attach it to the same item revisions as an existing view label, with a few additions or exceptions. The steps in this procedure explain how to create a view label based on an existing view label. For example, suppose builds are done only after a view has been rolled back to a label and that the build is given the same name as the label. If, in the last build, only one Help file was missing, you would probably change the existing label to include that one file and rebuild. However, if the previous build was already made available to users participating in a field test, using the same label could cause confusion. It would be better to create a new view label as a copy of the older label and then add the missing file to the new label.

Note: You cannot copy a view label unless it already exists in the view in which you are performing this operation. The view configuration must also be current.

1. Choose **View** > **Labels** . The **Labels** dialog box opens.
2. Click **New**. The **View Label** dialog box opens.
3. Type a name and description for the label. The maximum name length is 64 characters and the description length is 254 characters.
4. Select the **Labeled Configuration** option to attach the label to item revisions that have an existing label.
5. Optionally, uncheck **Use As Build Label** if you do not want this label to be a build label.

   Note: By default all view labels are designated as build labels.
6. Click **OK**.
7. Click **Close**. The new view label is now attached to the same revisions as the existing label.
8. Select the items in the upper pane for which the new label must differ.

   Tip: You can also select all items with a specific label. Right-click in the upper pane, choose **Select** > **By Label** . When you select the label, all the items attached to that label are automatically selected.
9. Detach the new label from items that you do not want to include.
10. Attach the new label to items formerly not included, and/or attach the new label to different revisions of items to which it is already attached.

Deleting Labels

In StarTeam, you can completely remove a view or revision label from a view, although you can create a new label with the same name later, if desired. When you delete a label, it is no longer visible in any list of labels, nor is it attached to any folders or items.

Note: If a label is frozen, you must unfreeze it before you can delete it.

1. Open the view for which you want to delete a label.
2. Choose **View** > **Labels** . The **Labels** dialog box opens.
3. Select the tab for the type of label you are deleting, the **View** tab for a view label, or the **Revision** tab for a revision label.
4. Select the label and click **Delete**. This removes the label from the view.

Detaching a Label from a Rolled-back View

Sometimes you need to detach a label from an item in a rolled-back view. For example, suppose you deleted a file that had view labels attached to it. Later you created a build label based on one of the view labels that was attached to the deleted file. If you roll back the view to the new build label in order to perform a build, the deleted file reappears in your view. If you do not want that file in this build, you can detach the new build label from that file. If you try to detach any other label from the rolled-back view, an error message informs you that you can detach only the label to which the view has been rolled back.

Note: You can attach and detach any labels from items in current view configurations, but you cannot see deleted items in those configurations. You can detach view labels from deleted items only if you roll back the view to a configuration based on the label you want to detach.

1. Click a component tab in the upper pane and select an item.
2. Click the **Label** tab in the lower pane.
3. Right-click the label you wish to remove and choose **Detach**.
4. Click **OK**.

The item from which the label is detached will disappear after a refresh.

Detaching a Label from a Specific Revision

If you decide not to include certain items in a view or revision label, you can detach the label from those items individually or as a group. Generally, the items from which labels are detached are files or folders.

To detach a label from a specific item revision

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab in the upper pane and select an item.
3. Click the **Label** tab in the lower pane.
4. Expand a revision in the **Label** pane to see all labels attached to it as children of the revision.
5. Right-click the label you wish to remove and choose **Detach**.
6. Click **OK**.

Detaching Labels from Folders

If you decide not to include certain folders in a view or revision label, you can detach the label from those folders.

Note: You can attach and detach any labels from items in current view configurations, but you cannot see deleted items in those configurations. You can detach view labels from deleted items only if you roll back the view to a configuration based on the label you want to detach.

To detach a view or revision label from a folder and its contents:

1. Right-click the folder in the **Folder Tree** or on the **Folder** tab and choose **Folder** > **Labels** to open the **Labels** dialog box. This dialog box lists the labels currently attached to this folder.
2. Select the label to be detached from the folder.
3. Click **Detach**.
4. Optionally, if you are detaching a revision label, select one of the following options:

   - **Folder Only**
   - **Folder And Items Contained In Folder**
   - **Everything In Subtree Rooted At Folder**

5. Click **OK**.

Note: When you detach a view label from a folder, the label is automatically detached from the items that the folder contains. It is also automatically detached from the child folders and their contents.

The folder from which the label is detached will disappear after a refresh.

Detaching Labels from Items

If you decide not to include certain items in a view or revision label, you can detach the label from those items individually or as a group. Generally, labels are detached from files or folders.

Note: You can attach and detach any labels from items in current view configurations, but you cannot see deleted items in those configurations. You can detach view labels from deleted items only if you roll back the view to a configuration based on the label you want to detach.

To detach a view or revision label from selected items

1. Select one or more items in the Server Explorer or in one of the Eclipse Explorers.
2. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
3. Click a component tab in the upper pane and select an item.
4. Right-click the selected items and choose **Labels** > **Detach** .

   The **Detach a Label** dialog box opens and displays all existing labels, identifying them as view or revision labels. By default, both **View Labels** and **Revision Labels** are checked.
5. Optionally, uncheck either **View Labels** or **Revision Labels** to limit the display list to a specific type of label.
6. Select a label from the list.
7. Click **OK**.

Freezing or Unfreezing Labels

When a label is frozen, the label cannot be:

- Attached to any additional folders or items.
- Detached from any folder or items.
- Moved from one revision of a folder or item to another.

Tip: You can identify a frozen label by a label icon containing a small snowflake displaying on a round, blue background. The icon displays in front of the label name in the list.

1. Choose **View** > **Labels** . The **Labels** dialog box opens.
2. Do one of the following:
   - Click the **View** tab if the label to be frozen is a view label.
   - Click the **Revision** tab if the label to be frozen is a revision label.
3. Select the label from the list.
4. Click **Freeze** or **Unfreeze**.
5. Click **Close**.

Reviewing and Moving Labels

To see the labels currently attached to specific items and folders, you can display them on the **Label** pane. In the case of revision labels, you can also move them to another revision of the folder or item.

Reviewing All Labels Attached to Item Revisions

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Select the item in the upper pane, then click the **Label** tab in the lower pane. This displays the **Label** pane which shows all revisions for the item.
4. Right-click a revision in the **Label** pane to display all of its labels.

Moving a Revision Label from One Item Revision to Another

1. Select a folder from the folder tree.
2. Click the component tab containing the items you want to label.
3. Select the item in the upper pane, then click the **Label** tab in the lower pane. This displays the **Label** pane which shows all revisions for the item.
4. Right-click a revision in the **Label** pane to display all of its labels.
5. Select a specific label and drag it from one revision to the another.

Reviewing the Labels Attached to a Folder's Revisions

1. Select a folder from the folder tree.
2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
3. The **Labels** dialog box lists all labels currently attached to this folder on a revision-by-revision basis.

Moving a Revision Label from one Folder Revision to Another

1. Select a folder from the folder tree.
2. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
3. Drag a revision label from one folder revision node to another in the **Labels** dialog box.

Sorting Labels Alphabetically

This topic describes how to sort labels alphabetically in descending or ascending order by the column name in the **Attach a Label** dialog box. It also explains how to change the ordering of the columns within the dialog box. The **Attach a Label** dialog box provide the following columns, all sortable and moveable for both view and revision labels:

- Name
- Description
- Time
- Creation On
- Created By

Note: The sorting and repositioning selections for the columns that you choose in the **Attach a Label** dialog box do not persist between sessions.

Sorting Labels for Folders

1. Right-click the folder and choose **Labels** to open the **Labels** dialog box.
2. Click **Attach**. The **Attach a Label** dialog box opens.
3. Click a desired column heading in the **Attach a Label** dialog box to sort. Click the column heading again to reverse the order.

Sorting Labels for Items

1. Right-click an item in the upper pane and choose **Labels** > **Attach** . This opens the **Attach a Label** dialog box.
2. Click a desired column heading in the **Attach a Label** dialog box to sort. Click the column heading again to reverse the order.

Branching

A branching view is a view that permits branching. This means that the folders and other items in the view can separate from the corresponding items in the parent.

Branching views serve many purposes. For example, you can create a branching view to:

- Meet different needs from those of your main line of development. For example, you might create a maintenance release or a custom version of your product, branched from a prior commercial release.
- Start development on the next release of your product by using some or all of the files from the previous release.
- Keep an area of your project private until it is completed and tested. Then you can merge your changes into the main line of development when and where necessary.

Only folders, files, and change requests can branch, although not every folder or every item in a branching view must branch. Requirements, tasks, and topics never branch.

Until an item branches, the corresponding items in both views remain identical. After an item branches, they are no longer identical, and the revision number indicates the new branch. The only way to make the items identical again is to manually merge them by comparing and merging views. After branching occurs, StarTeam no longer sends updates to nor applies updates from the corresponding item in the parent view.

For reasons of safety, deletions made in the parent view are not propagated to the child view and vice versa. If you want to delete a folder or item from all related views, you have to delete it manually from each of those views.

Also, a move is considered a copy operation followed by a delete operation. Consequently, the view in which the move was made has one copy of a folder or item in the new location, while the related views have two copies of the folder or item, one in the original location and one in the new location.

Note: Branching a view negates all shares, not just the ones between parent and child views.

Branching Options

Branching occurs when an item in the child view changes if its behavior is set to **Branch On Change**. When an item branches, a separation occurs between the item and its corresponding item in the parent view. These separate items also begin to have different branch revision numbers.

When creating a branching view, if you select:

Branch All The behavior of every item that is in the view at the time the view is created is set to **Branch On Change**.

Branch None The behavior of every item in the view at the time the view is created is not set to **Branch On Change**. Changes to any item with a floating configuration can be propagated to the parent view.

When you branch a view, any manual shares between items in the same view are not retained in the view's child view.

Note: Any item with a frozen or fixed configuration is read-only when its behavior is not set to **Branch On Change**. Read-only means that no data about this item within the view can be changed. For example, although you may be able to edit a file, you cannot check it in or change its properties.

As you add, move, share, and modify items, their behaviors can change.

Branching is Disabled

When the check box for **Branch On Change** is disabled the item cannot branch. One of the following is true:

- The item is original to the current view, not shared into it. In other words, it is the root item in its own reference tree.
- The item has already branched. (An item can branch only once per view.)

Branching Is Set to Branch On Change

When **Branch On Change** is both enabled and selected, branching occurs the next time the item changes. At that time, a separation occurs between the item in the new view and its corresponding item in the parent view. The item that becomes separated from its corresponding item in the parent view takes on the following behaviors:

- Its **Branch On Change** check box becomes disabled.
- Its revision number's dot notation expands to include two more numbers.

Branching Is Not Set to Branch On Change

When the **Branch On Change** check box is enabled but cleared, branching does not occur when you change the item.

If the item’s configuration floats, the change is propagated to the parent view.

If the item’s configuration does not float, the item cannot be changed because the parent view cannot be updated. The item is treated as though it were read-only. For example, if the item is a file, you can edit it but you cannot check it in or change its properties.

Branching Behavior of Items

Given the appropriate settings for folders, files, and change requests, you can branch these items in a child view—that is, you can separate these items from the corresponding items in the parent view.

Branching a folder does not branch its contents (child folders nor items.)

After an item branches, it receives a new revision number. For example, if a file's revision number (in dot notation) 1.13 before the file branches, it becomes 1.13.1.0 after branching. The next change to the file in the parent view will receive the revision number 1.14. The next change in the child becomes 1.13.1.1.

Below are the basic facts about branching behavior:

- Folders and change requests branch when their properties change.
- Files branch when either their contents or their properties change.
- Requirements, tasks, and topics can never branch.

Typical Branching Scenario

Suppose you are working on a product and a customer requests a special edition of the product with a few special features tailored specifically for that customer. To separate the current product's items from those for the special request, a branching view is created.

When items are branched, they are derived from other items that become their ancestors. Items may have several completely different revision histories with common ancestries. In the case of a text file, for example, the branched item can later be merged with the file from which it originated. For example, the development of a product for a new operating system may start with the existing files for the first operating system as its base.

History Affects Branching Behavior

Whether or not a folder, file, or change request has the ability to branch depends on its history. If you do not know the complete history, you should not assume that you know its behavior. For example,

- If a folder or item was in the parent view at the time the branching view was created, and if the branching view was created with **Branch All** as its branching option in the **New View Wizard**, the folder or item's branching behavior is initially enabled and the **Branch On Change** check box is selected in the **Folder Behavior** dialog box.
- If a folder or item was in the parent view at the time the branching view was created, and if the branching view was created with **Branch None** as its branching option, the folder or item's branching behavior is initially enabled and the **Branch On Change** check box is cleared. However, this behavior can be changed.
- If a folder or item is added to the branching view after the view is created, the folder or item's branching behavior is disabled. The **Branch On Change** check box is disabled and cleared in the **Folder Behavior** dialog box. However, if you share that folder or item, its branching behavior becomes enabled automatically in its new view.

Branching Behavior of Shared Items

The "branch on change" behavior of a shared item is specific to the folder it is in and the "branch on change" check box is selected by default for the shared file.

Creating a Branching View

A branching view is a new view derived from an existing view. When a branching view is created, StarTeam shares items from the parent view into the child view. The child view knows that the items came from the parent, but has no idea where the parent got them. For example, if a folder was shared to the parent view from another project, the child view does not have a similar share to the other project. If that is needed, the folder in the child view should be deleted and the folder should be shared to the child view from the other project (just as it was once shared to the parent view).

Similarly, if a folder was shared from the parent view to another project, or to another location within the parent view, the corresponding folder in the child view would not have that same relationship with the other project or view folder.

Note: Only folders, files, and change requests can branch. Requirements, tasks and topics never branch.

1. Display the project view upon which the new view will be based.
2. Choose **View** > **New** . The **New View Wizard** opens.
3. Select **Branch All** from the **View Type** list.
4. Type a **Name** and a **Description** for the view in the appropriate fields and click **Next**.
5. Select the **Root Folder** for the new view and click **Next**.
6. Type or browse for the name of an appropriate **Default Working Folder**.

   Caution: For a **Branch All** view, always use a working folder that is different from the one used by the parent view. Using the same working folder for the parent and child views can cause changes in one view to be overwritten when files are checked out from the other view. It can also result in incorrect or, at least, misleading file status indicators.
7. Click **Next** to display the **Configuration** page.
8. Select one of the available configuration options on the **Configuration** page.
9. Click **Finish**.

Effects on Change Requests When Branched, Moved, and Shared

The workflow of a change request may be significantly affected when the change request is moved, merged, or branches:

- If the **Last Build Tested** and the **Addressed In Build** fields have build labels as their values (if these fields are not empty and do not contain the value **Next Build**) the altered change request retains those values. In the new view, these values can be changed, but only to the names of build labels that exist in that view.
- If the **Addressed In Build** field contains the value **Next Build** at the time of the operation, this value is replaced by the name of the next build label created in the original view, not the next build label created in the new view. This action occurs even if other alterations have been made to the change request in the new view.
- If the **Last Build Tested** and the **Addressed In Build** fields have no values at the time of the operation, their workflow is specific to the view in which they currently reside.

Note: If a change request branches, its workflow is affected by its values in the **Last Build Tested** and the **Addressed In Build** fields at the time it branches.

Configuring a View to Display Non-Branched Files

If you use a branch-all variant view with item configurations set to a specific timestamp, you may periodically want to catch up the parent view with changes in the branched (child) view by performing a Re-base operation. While you can do this with View Compare/Merge, when a branch-all variant view is used as an **Activity** view, many files will not have been modified and hence will not have been branched. In this case, the non-branched files can be more quickly re-based with the main view by simply altering their configuration timestamp.

This procedure is a quick way to pull out all files that have not branched yet, and create a new view label to see them separately. This allows you to re-base the view on a controlled basis. The branched files, of course, have to be merged separately.

1. Right-click the column heading and choose **Sort and Group**.
2. Click **Show Advanced Fields** in the **Sort and Group** dialog box.
3. Change the sorting/grouping order so that the display groups by **Branch State**, followed by whatever else you normally like to see (such as sort by **Status**, then **Name**, and so on.)
4. Click **OK**.

   The non-branched files will be displayed in the upper pane.
5. Right-click the non-branched files and choose **Advanced** > **Behavior** .
6. Click the **Configuration** tab of the **Folder Behavior** dialog box.
7. Change the **Configuration Date** to a new timestamp, for example, the latest view label and click **OK**.

Tip: It is helpful to set up custom filters with the appropriate sort and group behaviors to make this easy.

Configuring the Branching Behavior of Shared Items

Folders and items can be shared from one view into another if both views belong to the same server configuration. They can also be shared from one folder to another within the same view. When you share folders and items that can branch, they acquire branching behavior in the new view. Requirements, tasks, and topics do not have branching behavior.

An item’s behavior determines whether the item branches on change. A shared item’s initial behavior in the new location depends upon the setting of the “Set items shared into view to branch on change” property when the item was shared. This view property appears on the **View Properties** dialog box in the root view and in branching views, but it does not appear in reference views. In reference views, the behavior of the shared folders and items that can branch depends on this property’s setting in the parent view of the reference view.

1. Choose **View** > **Properties** to open the **View Properties** dialog box.
2. Click the **Name** tab.
3. Check or clear the **Set Items Shared Into View To Branch on Change** check box.

   Note: Clearing the **Set Items Shared Into View To Branch on Change** check box is not recommended and will cause a warning message to be displayed upon saving.
4. Click **OK** or **Apply**.

After items have been shared into a view, you can change their behavior on an item-by- item basis, but additional changes to the **Set Items Shared Into View To Branch on Change** property do not change the behavior of the items.

Reviewing or Changing Branching Behavior

If a folder, file, or change request in a child view has the appropriate settings, you are able to branch it. Meaning, you can separate it from the corresponding item in the parent view.

At any time, you can determine whether a folder, file, or change request in a branch view is set to **Branch On Change** (that is, its current branching behavior), and if the**Branch On Change** field is enabled, you can change it's branching behavior.

Note: Folders and change requests branch when their properties change, while files branch when either their contents or their properties change. Requirements, tasks, and topics never branch.

1. Right-click a folder or item and choose **Advanced** > **Behavior** .
2. Select the **Modify** tab in the **Folder/ Item Behavior** dialog box, and view or modify the **Branch On Change** check box.

Personal Options

You and your team members can set personal options that suit your individual work styles. These options apply to the currently logged-on user on a given workstation. You can also update the list of servers available to your workstation and update your user account on the server on which you are currently logged in. Click **Tools** > **Personal Options** to adjust the way the following options work.

Note: If you have StarTeam Agile installed and configured to point to an existing StarTeam Server, you will also see **Sprint** and **Story** tabs.

Workspace Options

Tools > **Personal Options** > **Workspace**

The **Workspace** personal options allow you to select a variety of options that affect the way your workspace operates.

Confirm	**Delete**	Displays a **Confirm** dialog box for deletions.
	Move/Share	Displays a **Confirm** dialog box for move and share operations.
	Warnings	Displays a **Confirm** dialog box for warnings.

Display	**Toolbars**	Displays **Toolbars**.
	Status bar	Displays the **Status Bar**.
	Custom tools	Displays custom tools created as part of StarTeam Extensions. If no custom tools are configured, clear the **Custom tools** check box to prevent custom tools from attempting to load with each view window.
	Show history times as UTC	Displays time stamps in the Time column of the history pane in UTC times. UTC times end in “Z” to differentiate them from local times (Z stands for the "zero meridian", which goes through Greenwich, England). Displays time stamps in local time when unchecked.
	Sort view labels by name	Automatically sorts view labels alphabetically.
General	**Restore folder selection on tab change**	Returns StarTeam to the last folder that was selected for a specific component tab. If you have not selected a particular tab in this session, StarTeam automatically selects the root folder for the view.
	Reset scope to local on folder change	Resets the **All Descendants** button on the toolbar every time you change a folder. This saves you the time StarTeam would take to scan items. If unchecked, you must select the button manually.
	Maintain group state on folder/ scope change	Keeps your place in the upper pane even when you change folders or reset the **All Descendants** button. If unchecked, all groups collapse when you make a folder or scope change.
	Open URL in new window	The default setting for the way a view displays in your workspace when you open a URL link. When this check box is selected, any StarTeam view you open via a URL opens in a new window.
	Automatic refresh with maximum delay of ___ minutes	Specifies the maximum number of minutes between refreshes of your project. This refresh is the equivalent of pressing **Shift + F5** and updates the **Folder Tree** and the upper pane. It happens every X minutes unless an operation by the user such as pressing **Shift +F5** forces a refresh and resets the timer. Automatic refresh is designed to perform even if the client is minimized. When **Automatic refresh** enabled, if the network is down, **Automatic refresh** will attempt to refresh the data in the client resulting in a connection error.
	Restore shortcuts at startup	Reopen at startup any views that were left open when you exited the application the previous time.
	Enable File System Watching	If checked, then the operating system notifies the StarTeam Cross-Platform Client if files/folders are added, modified, or removed from the file system. If unchecked, then no such notification occurs.
	Sort open windows by name	If checked, sorts the opened windows by name. By default, windows are sorted by the order in which they are opened.
Reports	**Report output path**	Specifies the name and path to which your reports should be sent
StarTeam client log	**Log output path**	Specifies the name and path for your `StarTeam.Log` files. The `StarTeam.Log` file contains data about operations sent from your workstation to one or more servers, depending on what project views

you have open. The data includes the name of the project so that you can isolate data for a particular server when necessary.

Log errors

Records errors that occur while you are using the client application. The errors log lists the date and time you started your server configuration and any errors or failed operations between the server and client. The application identifies each failed operation by an internal ID and provides an explanation. For example, you might see: `...Operation 40956 failed: TCP/IP Socket Error 10054:...`. If you are logging both errors and operations, the application also logs the operation that the server was performing at the time of the error.

Log operations and events that take at least ___ (milliseconds)

Specifies that StarTeam should record file operations and/or events that take at least the specified number of milliseconds, and should send them to the `.log` files. The milliseconds time setting stops the log from filling up with operations and events of little importance. The default, 10 milliseconds, is a reasonable setting.

The **Summary** option includes a breakdown of the time spent on the client and the server for each operation, and the **Details** option lists the server commands along with the summary.

This option records information on the date, time, and UI Operation number for each command executed by your workstation. Operations can be executed on either the server or the client.

Log events

Specifies that StarTeamMPX events should be sent to the `.log` files. The log identifies the time and date on which a StarTeamMPX event (an automatic refresh or file status update) took place. The log prefaces a StarTeamMPX event as `Statistics for Events` and uses internal IDs and brief explanations to identify the server event.

The following example describes a status change for a file: `...Statistics for Events /1b21dd1-e208-51ea-01b2-1dd1e20851ea/Object/File/ Modify`.

You can log StarTeamMPX events only if you check **Enable MPX** on the **MPX** tab in the **Personal Options** dialog box. For StarTeamMPX related operations, any changes you make on the **Workspace** tab do not apply to projects already open. However, the application will log StarTeamMPX events for any projects you open after checking this option.

Advanced Options Button

Component Order for `ServerName`

Specifies the display order for the component tabs in the application. The tab display to the left of the **Component Order** button displays the tabs in the order you chose. Click **Component Order**.

Important: It is necessary to close and reopen the project for changes to the component tab order to take effect.

Note: Never list the **Audit component** tab first in the component list. Doing so greatly impacts performance.

Look and feel

Changes the look and feel of the application.

Note: This option is not available in the StarTeam Visual Studio Plugin.

Font adjustment

Increases or decreases the application font size.

Note: This option is not available in the StarTeam Visual Studio Plugin.

Application Instance

Specifies the default behavior for opening another instance of StarTeam when using a shortcut or URL.

Using a URL or shortcut defaults to using an open instance of the StarTeam Cross-Platform Client to open. However if the client has a dialog open inside some view (or some other modal operation in process), StarTeam cannot use open a project view in that client. In such situations, when you click a URL from email (for example) or a shortcut on your desktop, or a `.vcmx` file attached to a change package, a new instance of StarTeam is opened only if the option selected from the **Application Instance** group box allows it. If the option is not selected, and you try to open a shortcut, you may get an error saying your StarTeam Cross-Platform Client is busy.

Note: This option is not available in the StarTeam Visual Studio Plugin.

StarTeamMPX Options

Tools > **Personal Options** > **MPX**

Servers that use offer additional caching services and performance enhancements. To take advantage of these benefits, must be enabled and configured on your workstation so that any open project view can take advantage of .

The right end of the application status bar displays the current status of on your workstation. The words and icons on the status bar for are as follows:

Yellow lightning bolt

Indicates that MPX is available and enabled for the currently selected project view.

If Web Edition can use the default client profile for an MPX-enabled server and, therefore, take advantage of MPX, this icon appears in front of the server configuration's name in the browser window.

Gray lightning bolt

Indicates that MPX is available for the currently selected project view but that it has not been enabled in the client.

Red circle with a slash beside a yellow lightning bolt

Indicates that MPX is enabled for the currently selected project view, but something happened to break the connection. For example, the Message Broker may be stopped.

(no icon)

Indicates that MPX is not available for the currently selected project view.

Instant

Indicates that MPX's auto-refresh is turned on.

Auto

Indicates that your workstation's auto-refresh is turned on, but that MPX's auto-refresh is either turned off or unavailable. (Your workstation's auto-refresh option is on the **Workspace** tab of the **Personal Options** dialog box.)

Manual

Indicates that your workstation's auto-refresh is turned off and that MPX's auto-refresh is either turned off or unavailable. You must manually refresh the current project view by pressing `F5`.

Options

Enable

Enables your workstation to use StarTeamMPX if it is available on the server. Changing this check box does not affect open projects. StarTeamMPX is enabled by default.

Automatic refresh with

Enables automatic refresh of the application window by way of MPX, with options for setting the minimum and maximum delay times between refreshes. The default minimum is 30 seconds, and the default maximum is 0– seconds. If this option is unchecked, you must refresh manually (**Shift+F5** .)

When **Automatic Refresh** is enabled, after every change to the view, StarTeam waits a minimum number of seconds before refreshing. That means that if changes are infrequent, the application performs a refresh almost immediately. However, if changes 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.

Automatic refresh is designed to perform even if the client is minimized. When **Automatic refresh** enabled, if the network is down, **Automatic refresh** will attempt to refresh the data in the client resulting in a connection error.

Use MPX Cache Agent for

Use MPX Cache Agent for	**File Content**	Use the MPX Cache Agent to retrieve file content.
	Item Properties	Use the MPX Cache Agent to retrieve item properties.
	Use MPX Cache Agent at	Designates a specific MPX Cache Agent to use by IP address and port number.
	Automatically locate the closest MPX Cache Agent for Server	Locates the network nearest MPX Cache Agent automatically, but only if the server is MPX-enabled.
	Maximum request threads	Specifies the maximum number of request threads allowed. The default is 2, and 2 to 3 should be adequate for most of your needs.

File Options

Tools > **Personal Options** > **File**

Use **File Options** to customize the way you work with files. In a few cases (such as Marking Unlocked Files Read Only), your administrator's choices may override your preferences.

Check-out	**Use last modification time for check-out files**	Uses the same time for each checked-out file as the time stamp of the revision being checked out. Otherwise, the time stamp used for the checked-out file is the current time (the time check-out occurs.)
Merging	**Always pop-up merge utility**	Opens the merge utility to display the merged file even when there are no conflicts.
	Pop-up merge utility in case of conflicts only	Opens the merge utility only if the merged file contains conflicts. If unchecked, files will be checked in and out automatically.

Deleted Status Values

Show 'Deleted' File Status Values Shows the status values for deleted files.

Locking

Exclusively lock files on check-out Sets the default **Lock Status** option to **Exclusive** in the **Check Out** dialog box . Otherwise, the default is **Keep Current**.

Clear file locks on check-in Sets the default **Lock Status** option to **Unlock** in the **Check Out** dialog box. Otherwise, the default is **Keep Current**.

Use non-exclusive locks in integration Creates a non-exclusive lock when locking a file from the application integration– that is, a lock that allows others to check in the file. Using non-exclusive locks also allows more than one person to edit a file at one time. If team members are not editing the same lines of the file, the merged file usually has no conflicts.

Mark unlocked working files read-only Sets working copies of files that you have not locked to read-only when you add files, check in files, check out files, or unlock files. If this check box is selected, only locked files can be edited.

EOL

Automatic EOL conversion for check-out operations Performs an automatic EOL conversion on check-out operations. If checked, also select the operating system on which you are working:

- Windows `(CR-LF)`
- Unix`(LF)`
- Mac`(R)`

Files can be checked out in **LF** format on every platform, regardless of specific options.

The EOL Format property can be set in the StarTeam Cross-Platform Client in the **Add/Check-in** and **File Properties** dialog boxes.

The default for automatic EOL conversion for check-out operations is “checked” if the user does not have that option defined already.

The EOL Property values are:

Undefined	(null in the SDK).
Client Defined	Causes workstation default or per-checkout EOL conversion option to be used.
Fixed CR, Fixed LF, and Fixed CRLF	Causes this EOL format to be used always. The workstation/check-out conversion option is ignored.

Note: Once EOL Format is defined, **Update Status** works for all text files, regardless of what EOL format was used when they were checked-out. For compatibility with older Clients, if check-out "EOL conversion" is not requested, and EOL Format is Undefined, files are still checked out with the EOL convention with which they were added to the StarTeam Server.

General

Use file checksums (MD5) to calculate status Uses the checksum instead of the file time stamp and size to compute the **Status** field when the application is refreshed. Using the checksum provides a more accurate status value than the time stamp, but takes longer. If unchecked, the application uses the time stamp and size.

File encoding for keyword expansion Specifies the code page to be used for keyword expansion by choosing a default file encoding from the list.

Repository

File status repository default Indicates where you want file status information stored, either in a central repository location on your workstation or in a child folder (named `.sbas)` of each working folder.

Central You can enter or browse for a location on your computer other than the default central repository location. Whenever you make a change to a file in the working folder, the status for that file is undated only on your computer in the specified location. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files.

Per-folder Useful in the special case where multiple users are sharing a working folder, for example, on a shared network drive. For example, suppose several users all check files in and out of a shared working folder. If these users have set the central repository option for file statuses, the statuses are stored on each of their computers. Whenever a user makes a change to a file in the working folder, the status for that file is undated only on that user’s computer. Everyone else sees the status **Unknown** for that file. Over time, all the files may have been changed, and the statuses can become **Unknown** for all users of all files. Using the per-folder option causes the statuses to be updated within the working folder itself. Everyone has access to those status changes and **Unknown** statuses do not occur.

Purge Opens the **Status Repository Cleanup** dialog box where you can remove file status data from the workstation status repository.

Default Resets the **Central** repository location to the default setting

URL Options

Display template Specifies a special template used to generate an HTML representation of an item when the item’s URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `~~FolderPath~~:~~Name~~`, the HTML representation will be the path to the selected file: `StarTeam\:buildinfo.properties`. This template is a super-set of that used by the report feature of the client.

Generate ID-based URLs Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:`

`49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`.

Note: Folders always use an ID-based URL.

Alternate Applications Opens the **Alternate Applications** dialog box where you can specify an alternate editor, merge utility, and comparison utility to use in the application if you don't want to use the default tools for those functions. Includes fields for specifying options to use with the applications.

Open With... Enables you to provide a command on a non-Microsoft Windows system that will display at least one type of files and folders. The command should consist of the path to an application and the command-line options for which the application for which the application can substitute the selected file. The application runs this command whenever you do one of the following: Double-click a file or folder in the item list, double-click an attachment, or generate and open a report.

The following command is suggested: `netscape -remote "openFile($file)"` because Netscape can handle many different media types, such as image files, text files, and HTML.

Merge Utility Options Use the following command-line options to represent files sent to the alternate merge utility.

$branchtip A place holder for the path to the tip revision of the file to be merged.

$usertip A place holder for the path to the local working file to be merged.

$basefile A place holder for the path to the common ancestor for the `$branchtip` and `$usertip` files.

$resultfile A place holder for the path to the file that will store the output from the merged file.

Compare Utility Options Use the following command-line options to represent files sent to the alternate compare utility.

$file1 A place holder for the path to the first of the two files to be compared.

$file2 A place holder for the path to the second of the two files to be compared.

Folder Options

Tools > **Personal Options** > **Folder**

The folder options enable you to specify the criteria that the application uses to determine whether a folder has been read. You can also indicate how often the application should search for new folders and how folder locking issues should be handled.

Check-out

Use Last Modification Time for Folders During Checkout Uses the last modification time when Working folders are created. Otherwise it uses the current time.

Locking

Exclusively Lock Folder During Edit Locks an asset when you open its **Properties** dialog box for editing. If unchecked, the application does not lock assets when you open its **Properties** dialog box.

Clear Manually Locked Folders After Edit Unlocks a locked asset after you have edited its properties and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks.

Note: If you do not select either of the locking options, opening a change request will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, change requests that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear change request Locks** option only, any change request that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock change requests manually or by opening them. These change requests become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

URL Options

Display template Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `~~FolderPath~~:~~Name~~`, the HTML representation will be the StarTeam path to the selected folder: `SampleProject\ReadMe`. This template is a super-set of that used by the Report feature of the client.

Generate ID-based URLs Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`.

Note: Folders always use an ID-based URL.

Show Not-in-View Folders By Default Checks the **Show Not-in-View Folders** on the **Folder Tree** menu to set it to be on by default. (Changing this check box does not affect projects that are already open.)

Change Request Options

Tools > **Personal Options** > **Change Request**

Use the **Change Request** options to specify the criteria that the application uses to determine whether a change request has been read. You can also indicate how often the application should search for new change requests and how change request locking issues should be handled.

Mark as read

When change request is selected Marks an asset as read as soon as you select it. An unread asset is displayed with bold font. A read asset is displayed with regular font.

	When selected for ___ seconds	Marks an asset read after it has been selected for the specified number of seconds. The range is from 15 to 9999 seconds.
	Only when manually marked as read	Marks a selected asset read when you choose **Change Request** > **Mark as Read** . **Note:** Assets are always marked as read when you display their properties.
System tray notifications	**Check for new or modified change requests**	Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.
	Interval (in minutes)	Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.
Locking	**Exclusively lock change request during edit**	Locks an asset when you open its **Properties** dialog box for editing. If unchecked, the application does not lock assets when you open its **Properties** dialog box.
	Clear manually locked change requests after edit	Unlocks a locked asset after you have edited its properties and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks.
URL Options	**Display template**	Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `Change Request:~~CR Number~~:~~CreatedBy~~`, the HTML representation will be the `Change Request:38,849:Tom Smith`. This template is a superset of that used by the Report feature of the client.
	Generate ID-based URLs	Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`. **Note:** Folders always use an ID-based URL.

Note: If you do not select either of the locking options, opening a change request will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, change requests that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear change request Locks** option only, any change request that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock change requests manually or by opening them. These change requests become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

Requirements Options

Tools > **Personal Options** > **Requirement**

Use the **Requirement Options** to specify the criteria that the application uses to determine whether a requirement has been read. You can also indicate how often the application should search for new requirements and how requirement locking issues should be handled.

Mark as read	**When requirement is selected**	Marks an asset as read as soon as you select it. An unread asset is displayed with bold font. A read asset is displayed with regular font.
	When selected for ___ seconds	Marks an asset read after it has been selected for the specified number of seconds. The range is from 15 to 9999 seconds.
	Only when manually marked as read	Marks a selected requirement read when you choose **Requirement** > **Mark as Read** . **Note:** Assets are always marked as read when you display their properties.
System tray notifications	**Check for new or modified requirements**	Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.
	Interval (in minutes)	Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.
Locking	**Exclusively lock requirement during edit**	Locks an asset when you open its **Properties** dialog box for editing. If unchecked, the application does not lock assets when you open its **Properties** dialog box.
	Clear manually locked requirements after edit	Unlocks a locked asset after you have edited its properties and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks. **Note:** If you do not select either of the locking options, opening a change request will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, change requests that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear change request Locks** option only, any change request that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock change requests manually or by opening them. These change requests become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

URL Options	**Display template**	Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `Requirement #~~Number~~: ~~Status~~`, the HTML representation will be `Requirement #34,132: Submitted`. This template is a superset of that used by the Report feature of the client.
	Generate ID-based URLs	Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`. **Note:** Folders always use an ID-based URL.

Task Options

Tools > **Personal Options** > **Task**

Use the **Task Options** to specify the criteria that the application uses to determine whether a task has been read. You can also indicate how often the application should search for new tasks and how task locking issues should be handled.

Mark as read	**When task is selected**	Marks an asset as read as soon as you select it. An unread asset is displayed with bold font. A read asset is displayed with regular font.
	When selected for ___ seconds	Marks an asset read after it has been selected for the specified number of seconds. The range is from 15 to 9999 seconds.
	Only when manually marked as read	Marks a selected task read when you choose **task** > **Mark as Read** . **Note:** Assets are always marked as read when you display their properties.
System tray notifications	**Check for new or modified tasks**	Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.
	Interval (in minutes)	Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.
Locking	**Exclusively lock task during edit**	Locks an asset when you open its **Properties** dialog box for editing. If unchecked, the application does not lock assets when you open its **Properties** dialog box.

Clear manually locked tasks after edit

Unlocks a locked asset after you have edited its properties and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks.

Note: If you do not select either of the locking options, opening a change request will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, change requests that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear change request Locks** option only, any change request that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock change requests manually or by opening them. These change requests become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

URL Options

Display template

Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `Task #~~Task Number~~:~~Status~~:~~Responsibility~~`, the HTML representation will be `Task #1,456:Ready To Start:Tom Smith`. This template is a super-set of that used by the Report feature of the client.

Generate ID-based URLs

Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`.

Note: Folders always use an ID-based URL.

Topic Options

Tools > **Personal Options** > **Topic**

Use the **Topic Options** to specify the criteria that the application uses to determine whether a topic has been read. You can also indicate how often the application should search for new topics and how topic locking issues should be handled.

Mark as read

When topic is selected

Marks an asset as read as soon as you select it. An unread asset is displayed with bold font. A read asset is displayed with regular font.

When selected for ___ seconds

Marks an asset read after it has been selected for the specified number of seconds. The range is from 15 to 9999 seconds.

Only when manually marked as read

Marks a selected topic read when you choose **topic** > **Mark as Read** .

Note: Assets are always marked as read when you display their properties.

System tray notifications

Check for new or modified topics
Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.

Interval (in minutes)
Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.

Locking

Exclusively lock topic during edit
Locks an asset when you open its **Properties** dialog box for editing. If unchecked, the application does not lock assets when you open its **Properties** dialog box.

Clear manually locked topics after edit
Unlocks a locked asset after you have edited its properties and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks.

Note: If you do not select either of the locking options, opening a change request will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, change requests that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear change request Locks** option only, any change request that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock change requests manually or by opening them. These change requests become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

URL Options

Display template
Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a topic: `Topic # ~~topicnumber~~: ~~Title~~, Status - ~~status~~`, the HTML representation will be `Topic #34,132: Topic Title, Status - Active`. This template is a super-set of that used by the Report feature of the client.

Generate ID-based URLs
Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`.

Note: Folders always use an ID-based URL.

Sprint Options

Tools > **Personal Options** > **Sprint**

Use the **Sprint** options to specify when the application looks for new or modified sprints.

System tray notification	**Check for new or modified sprints**	Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.
	Interval (in minutes)	Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.
URL Options	**Display template**	Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a sprint: `sprint # ~~sprintnumber~~: ~~Title~~, Status - ~~status~~`, the HTML representation will be `Sprint #34,132: Sprint Title, Status - Active`. This template is a super-set of that used by the report feature of the client.
Generate ID-based URLs	Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`. **Note:** Folders always use an ID-based URL.	

Story Options

Tools > **Personal Options** > **Story**

Use the **Story** options to specify when the application looks for new or modified stories.

System tray notification	**Check for new or modified stores**	Checks for new or modified assets at regular intervals and lets you know that you have become responsible for new assets. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new change request.
	Interval (in minutes)	Specifies the number of minutes between automatic checks for new or modified assets. The default is 10 minutes.
URL Options	**Display template**	Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a story: `story`

`# ~~storynumber~~: ~~Title~~, Status - ~~status~~`, the HTML representation will be `Story #34,132: Story Title, Status - Active`. This template is a superset of that used by the report feature of the client.

Generate ID_based URLs Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a name-based URL would be `starteam://hostname:49201/myproject`.

Note: Folders always use an ID-based URL.

Custom Component Options (Personal Options Dialog Box)

Tools > **Personal Options** > **Custom Component**

The custom component options enable you to specify the criteria that the application uses to determine whether a component has been read. You can also indicate how often the application should search for new components and how component locking issues should be handled.

Mark as read

When <custom component> is selected Marks a <custom component> read as soon as you select it. An unread <custom component> is displayed with bold font. A read custom item is displayed with regular font.

When selected for ___ seconds Marks a <custom component> read after it has been selected for the specified number of seconds. The range is from 15 to 9999 seconds.

Only when manually marked as read Marks a selected <custom component> read when you choose **Custom** > **Mark as Read** .

Note: Custom items are always marked as read when you display their properties.

System tray notifications

Check for new or modified custom items Checks for new or modified <custom component> at regular intervals and lets you know that you have become responsible for new <custom component>. If this option is checked, you must also specify the number of minutes between checks in the Interval option below. When unchecked, the application does not place an icon in the system tray for a new custom item.

Interval (in minutes) Specifies the number of minutes between automatic checks for new or modified <custom component>. The default is 10 minutes.

Locking

Exclusively lock <custom component> during edit Locks a <custom component> when you open its **Properties** dialog box for editing. If unchecked, the application does not lock <custom component> when you open its **Properties** dialog box.

Clear manually locked <custom component> after edit Unlocks a locked <custom component> after you have edited its properties, and clicked **OK** to create a new revision. If unchecked, the application does not remove the locks.

URL Options	**Display template**	Specifies a special template used to generate an HTML representation of an item when the item's URL is copied to the Clipboard. With no format, there is a default HTML representation that specifies the type of item and identifies it by name and number, for example, `Custom Item #39,849`. When the text is generated from the template, the specified property values are substituted for the variables in `~~*~~`. The variables may be referenced by the same names used in report templates, as well as by the display name of the property. When using the display name, you can omit spaces, and case will be ignored. For example, if you use the following sample template for a file: `Custom Item:~~CI Number~~:~~CreatedBy~~`, the HTML representation will be the `Custom Item:38,849:Tom Smith`. This template is a superset of that used by the Report feature of the client.
	Generate ID-based URLs	Specifies the URL by ID rather than by name. For example, an ID-based URL would be `starteam://hostname:49201/12;ns=Project;scheme=id`, while a namebased URL would be `starteam://hostname:49201/myproject`. **Note:** Folders always use an ID-based URL.

If you do not select either of the locking options, opening a custom item will not lock it; you must manually lock and unlock it. If you select the **Exclusively Lock** option only, custom items that are not already locked become locked when you open them and unlocked when you click **Cancel** or **OK**. If you select the **Clear <custom component> Locks** option only, any custom component that you have locked manually becomes unlocked when you click **OK** to create a new revision. If you select both options, you can lock custom components manually or by opening them. These custom components become unlocked when you click **OK** to create new revisions or (if they were not locked prior to being opened) when you click **Cancel**.

Audit Log

The audit log is a record of events that happen to your assets. Click the **Audit** tab to display audit log entries for the selected view.

Audit Log Events

Events are actions performed on an owner. For example, a file can be checked in or removed from version control. Such events are recorded in the audit log. Most items can be:

- Added
- Branched
- Comment Edited
- Created
- Deleted
- Locked
- Lock Broken
- Modified
- Moved From
- Moved to
- Shared
- Unlocked
- Converted

- Edited
- Item Overwritten (as foreign archive files become native files)
- Vault
- Created
- Modified
- Deleted
- Frozen
- Unfrozen
- Attached
- Moved
- Detached
- Modified

Filtering Audit Log Entries

When you click the **Audit** tab, it displays audit log entries for the selected view in the upper pane of the project view window. The **Audit** menu item also becomes available on the menu bar. The list of audit records depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or **Audit** menu. If you select this tab and the upper pane is empty, your administrator has disabled the audit log.

Filtering allows you to specify what fields are displayed in the audit entries that appear in the upper pane and how those fields are grouped and sorted.

Note: You can customize the default filters or create new ones. You can also limit the number of audit log entries displayed by creating a query that selects audit log entries by specific property values.

1. Click the **Filter** list on the tool bar.
2. Select one of the following default filters:

<By Class and Event>	Displays audit entries sorted by their value in the Class Name 1 field (type of item) and Event (type of action) field.
By Transaction and Event	Groups audit log entries by descending **Transaction ID** and then by **Event** type. This filter provides a reverse-chronological list of updates in the view by transaction.
Events	Groups audit log entries by **Event** type, then by **Target 1 Class ID**, and then by **Created Time**.
Show All	Displays all entries. This is the default option.

Searching for Log Entries

When you click the **Audit** tab, it displays audit log entries for the selected view in the upper pane of the project view window. The **Audit** menu item also becomes available on the menu bar. The list of audit records depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or **Audit** menu. If you select this tab and the upper pane is empty, your administrator has disabled the audit log.

All entries shown in the **Audit** list:

- Are associated with the folder selected from the StarTeam folder tree.
- Match the filter selected from the **Filter** list.
- Match the depth specified by the **All Descendant** button.

Do one of the following:

1. Click on **Audit** > **Find**
2. Click on **Audit** > **Find Next**
3. Click on **Audit** > **Find Previous**

Sorting Audit Log Entries

The sort usually takes place in descending or ascending numeric or alphanumeric order depending on the data.

1. Click on an audit column header to perform a sort based on the value in that column.
2. To change the sort order from ascending to descending or vice versa, click the header a second time.

Sending Log Entries Through E-mail

You can send an audit list item as an e-mail, although the attachments in the item will not be sent.

1. Select the item you want to send by clicking on it.
2. Click **Audit** > **Send To** . The **Send To** dialog box opens allowing you to send the selected item in the **Audit** list as an e-mail.

Audit Fields

This section lists all the audit fields in alphabetical order.

Class Name 1

Values: text

Internal Identifier: `Class Name 1` (contains spaces)

The name of the class of items, such as Label, Promotion State, Folder, File, Change Request, Topic, Task, or Trace.

Class Name 2

Values: text

Internal Identifier: `Class Name 2` (contains spaces)

The name of the class of items, such as Folder, File, Change Request, Label, Topic, Task, or Trace.

Class Name 3

Values: text

Internal Identifier: `Class Name 3` (contains spaces)

The name of the class of items, such as Folder, File, Change Request, Label, Topic, Task, or Trace.

Created By

Values: list of users, `<None>`

Internal Identifier: `CreatedUserID`

Always empty because the audit entry is created by the system.

Created Time

Values: date/time

Internal Identifier: `CreatedTime`

The time at which this entry was created.

Deleted By

Values: list of users, `<None>`

Internal Identifier: `DeletedUserID`

The name of the user who deleted an audit entry. Because deleted entries do not appear in the list, this information is unavailable to users.

Deleted Time

Values: date/time

Internal Identifier: `DeletedTime`

The time at which an audit entry was deleted. Because deleted entries do not appear in the list, this information is unavailable to users.

Event

Values: `Added, Branched, Comment Edited, Created, Deleted, Edited, Item Overwritten, Label Attached, Label Created, Label Deleted, Label Detached, Label Frozen, Label Modified, Label Moved, Label Unfrozen, Lock Broken, Locked, Modified, Moved From, Moved To, Promotion Model Modified, Promotion State Modified, Shared, Unlocked, Vault Converted`

Internal Identifier: `EventID`

The name of the operation being recorded.

Folder

Values: text

Internal Identifier: `Folder`

The name of the folder that stores the audit entry.

Folder Path

Values: text

Internal Identifier: `Folder Path` (contains spaces)

The path to the folder that stores the audit entry.

Folder VMID (Advanced)

Values: number

Internal Identifier: `FolderVMID`

The ID assigned to the folder that stores the item.

Item 1

Values: text

Internal Identifier: `Item 1` (contains spaces)

Indicates what class `1` item received the audited operation. This can be the name of a file or task, the number of a change request or requirement, or the title of a topic.

Item 1 Info

Values: text

Internal Identifier: `Info`

Provides the revision number in dot notation for the class `1` item, if it is revisionable. For example, a label can be a class `1` item and it does not have revisions.

Item 2

Values: text

Internal Identifier: `Item 2` (contains spaces)

Indicates what class 2 item received the audited operation. For example, if a label was attached to a file, the class `1` item is the label and the class 2 item is the file.

Item 2 Info

Values: text

Internal Identifier: `Info2`

Provides the revision number in dot notation for the class 2 item, if it is revisionable. For example, a label can be a class 2 item and it does not have revisions**.**

Item 3

Values: text

Internal Identifier: `Item 3` (contains spaces)

Indicates what class 3 item received the audited operation. For example, if a label was moved from one revision to a file to another, the class `1` item is the label, the class 2 item is the revision of the file that was initially , and the class 3 item is the final revision of the file.

Item 3 Info

Values: text

Internal Identifier: `Info3`

Provides the revision number in dot notation for the class 3 item, if it is revisionable. For example, a label can be a class `1` item and it does not have revisions.

Modified By

Values: list of users, `<None>`

Internal Identifier: `ModifiedUserID`

Does not apply to audit entries.

Modified Time

Values: date/time

Internal Identifier: `ModifiedTime`

Does not apply to audit entries.

Object ID

Values: number

Internal Identifier: `ID`

Each audit entry is assigned an object ID when it is added to a view.

Project

Values: list of projects in this server configuration, `<None>`

Internal Identifier: `ProjectID`

The name of the project in which an audit entry was recorded.

Target 1 Class ID (Advanced)

Values: number

Internal Identifier: `Target 1 Class ID` (contains spaces)

The ID number assigned to class `1` items or a `-1` if there is no ID.

Target 1 Object ID (Advanced)

Values: number

Internal Identifier: `Target 1 Object ID` (contains spaces)

The object ID for the class `1` item that received the audited operation or a `-1` if there is no ID.

Target 1 Revision Time

Values: date/time

Internal Identifier: `Target 1 Revision Time` (contains spaces)

The time at which the last revision was made to the class 1 item that received the audit operation.

Target 2 Class ID (Advanced)

Values: number

Internal Identifier: `Target 2 Class ID` (contains spaces)

The ID number assigned to class 2 items or a `-1` if there is no ID.

Target 2 Object ID (Advanced)

Values: number

Internal Identifier: `Target 2 Object ID` (contains spaces)

The object ID for the class 2 item that received the audited operation or a `-1` if there is no ID.

Target 2 Revision Time

Values: number

Internal Identifier: `Target 2 Revision Time` (contains spaces)

The time at which the last revision was made to the class 2 item that received the audit operation.

Target 3 Class ID (Advanced)

Values: number

Internal Identifier: `Target 3 Class ID` (contains spaces)

The ID number assigned to class 3 items or a `-1` if there is no ID.

Target 3 Object ID (Advanced)

Values: number

Internal Identifier: `Target 3 Object ID` (contains spaces)

The object ID for the class 3 item that received the audited operation or a `-1` if there is no ID.

Target 3 Revision Time

Values: date/time

Internal Identifier: `Target 3 Revision Time` (contains spaces)

The time at which the last revision was made to the class 3 item that received the audit operation.

Transaction ID (Advanced)

Values: number

Internal Identifier: `TransactionID`

Uniquely identifies the database transaction that contained the update represented by the audit record. (A database transaction can contain multiple updates.) Note that audit records created before the database was upgraded to a StarTeam release that records a **Transaction ID** will have a **Transaction ID** of `-1`.

User

Values: list of users, `<None>`

Internal Identifier: `UserID`

The name of the user who performed the recorded operation.

View

Values: list of views, `<None>`

Internal Identifier: `ViewID`

The name of the view in which an audit entry was recorded.

Queries

You can use a query to limit the items that display. Each query is performed on all items in the StarTeam folder and component you have selected. The fields included in the query do not have to display. Once a query has been created, it can be used in every project in the same server configuration.

StarTeam queries have the following attributes:

- A unique name that easily identifies the query. Query names are not case-sensitive.
- Public or private status. Anyone with appropriate access rights can use public queries, while private queries are available only to your user ID. Once a query has been saved with a specific status, its status cannot be changed. However, you can copy a query and change the state of the new query.
- A logical expression appropriate for items of a particular type. These expressions include one or more conditions. A condition consists of a field (not necessarily a current column header), a relational operator, and a value to be compared to the value of the field. For example, a condition used to locate change requests might be: `Responsibility Equals Rhonda Thurman`. More complex queries include two or more conditions bound together by logical operators: `AND`, `OR`, and `NOT`. For example, to locate all the change requests for which `Rhonda Thurman` is responsible that also have a `high` severity, use: `Responsibility Equals Rhonda Thurman AND Severity Equals High`.

Note: If you are creating a complex query, and the first logical operator in your query should be `OR`, select the `AND` logical operator in the query tree. Then click the **AND->OR->NOT** button. This changes an `AND` to an `OR`. Similarly, one more click changes the `OR` to a `NOT`. Keep toggling the button until the operator that appears is the one you want to use. It is best to use the condition or logical operation that will result in the fewest matches as the first condition or logical operation.

Creating Queries

You can write simple queries that have only one condition, or complex queries that use several conditions and one or more logical operators.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Click **New**. The **New Query** dialog box opens.
5. Type a name for the query in the **Query Name** field.
6. Select the **Public** check box to add this query to the project (and the server configuration), allowing anyone with the appropriate access rights to use it. If you do not check the **Public** check box, the query will be private, that is, available only to your user ID.
7. Select a **Field** and **Operator**, type or select a **Value**, and click **Add** to place this condition in the **Query** tree.

   By default, the **Query** tree contains the **AND** operator as the root of the tree, which you cannot delete. If there is only one condition, StarTeam ignores the logical operator.
8. Click **View as Text** to view the query in text format. Notice that the default logical **AND** operator was not included in your query. Click **OK**.
9. Optionally, click one of the following **Logical Node** buttons to create a new **Query** tree node: **AND**, **OR**, or **NOT**.

   Tip: You can change an existing operator in a condition by toggling the **AND->OR->NOT** button. Keep clicking the button until the operator that appears is the one you want to use.
10. Select the fields for this new condition and click **Add**.
11. Add any other conditions, then click **Save**.

    The **Queries** dialog box now contains your new query enabling you to select it for querying data.

    Note: If this is a public query, you might want to set access rights for it.

Tip: When creating a query condition, it is best to use the condition or logical operation that will result in the fewest matches as the first condition or logical operation.

Creating "Me" Queries

StarTeam has the capability of creating "Me" queries that allow a query to be set up which is evaluated against the currently logged in user ("Me"), rather than having to specify a specific username at the time of query creation.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Click **New**. The **New Query** dialog box opens.
5. Type a name for the query in the **Query Name** field.
6. Select the **Public** check box to add this query to the project (and the server configuration), allowing anyone with the appropriate access rights to use it. If you do not check the **Public** check box, the query will be private, that is, available only to your user ID.
7. Select **Created By** for the **Field** in the **Condition Node** area. Select **Equals** for the **Operator**, and select `Me` in the **Value** list. Click **Add** to place this condition in the **Query** tree.

   By default, the **Query** tree contains the **AND** operator as the root of the tree, which you cannot delete. If there is only one condition, StarTeam ignores the logical operator.
8. Click **Save**. The **Queries** dialog box now contains your new query enabling you to select it for querying data.

Applying Queries

Once a query has been created, you can use it to:

- List only those items that match the query in the upper pane.
- Select items from the upper pane that match the query (even though other items are still displayed).

Note: Using a query allows you to treat items as a group. For example, you can print a report on them, attach a label to them, check them out, and so on.

Applying an Existing Query to Items in the Upper Pane

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Select a query from the list.
5. Click **Select** to apply the query to the items shown on the upper pane. This action changes the contents of the upper pane by displaying only those items that meet the specifications of the query.
6. If you are not satisfied with the results of the query, reopen the **Queries** dialog box, and click **Clear Query** to return to the previously displayed list of items.

Selecting Items in the Upper Pane that Match an Existing Query

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Select a query from the list.
5. Click **OK**. This highlights the items in the upper pane that match this query.
6. To deselect the query items, click the upper pane.

Queries Options

Component > **Filters** > **Queries**

Use the **Queries** dialog box to view and apply the currently defined queries.

Note: You can also access the **Queries** dialog box from the context menu by right-clicking on a column title in any component.

Queries list	Lists all existing queries for this component. Note that the multi-person icon left of the query name indicates a public query; a single-person icon indicates a private query.
Close button	Closes the **Queries** dialog box.
Select button	Applies the selected query to the data listed in the selected component.
Clear Query button	Clears the current query and displays all data.
New button	Opens the **New Query** dialog box for you to define a new query.
Edit button	Opens the selected query definition in the **Edit Query** dialog box for you to edit.
Copy button	Opens the **Copy Query** dialog box for you to enter a name for your new query.
Delete button	Deletes the selected query definition.
Access Rights button	Allows you to assign access rights to a query.

New Query Options

Component > **Filters** > **Queries**

Click **New** to define a new query in the **New Query** dialog box.

Note: You can also access the **Queries** dialog box from the context menu by right-clicking on a column title in any component.

Query Displays the query definition parameters.

Name	Enter a unique name for the new query.
Public Check Box	Check to give the query public status. Public queries can be used by anyone with the appropriate access rights, while private queries are available only to your user ID. Once a query has been saved with a specific status, its status cannot be changed. However, you can copy a query and change the state of the new query.
Query List	Displays the query definition. By default, the `AND` condition appears as a starting logical condition.

Logical node Use to choose the logical operator for each condition.

AND	Click to add an `AND` condition to the query.
OR	Click to add an `OR` condition to the query.
NOT	Click to add a `NOT` condition to the query.
AND->OR->NOT	Click to toggle the default `AND` condition to an `OR` or `NOT` condition.

	Remove	Click to delete the currently selected condition. You must confirm the deletion.
Condition Node	Use to define the conditions for the logical operator.	
	Field	Lists all the fields available for this component.
	Operator	Lists all of the operators that can be specified for the selected field.
	Value	Use to specify a value for this field and condition.
	Show advanced fields check box	Select to display all possible fields including the advanced fields.
	Show deleted users check box (optional)	For components with user fields only (such as the Change Requests component), select to show deleted users in the query results.
	Alphabetical check box (optional)	For enumerated fields only (fields that have specified values), select to alphabetize the query results rather than list them in the order in which they appear in the enumeration list.
	Add	Click to add the condition to the query definition.
	Modify	Click to modify the selected query condition.
	Delete	Click to delete the selected query condition.
View as Text button	Displays the current query definition in a field.	
Save button	Saves the query.	
Cancel button	Cancels the query definition.	

Relational Operators Used in Queries

The relational operators that you can use to define conditions in a query vary according to the type of field:

- Text fields
- Boolean, enumerated type, and numeric fields
- Date/time fields

Relational Operators Used on Text Fields

The relational operators that can be used on text fields are:

- Equals
- Is Not
- Contains (ignore case)
- Contains (match case)
- Starts with (ignore case)
- Starts with (match case)
- Ends with (ignore case)
- Ends with (match case)

Relational Operators Used on Boolean, Enumerated Type, and Numeric fields

The relational operators that can be used on Boolean, enumerated type, and numeric fields are:

- Less Than
- Same or Less
- Equals
- Same or Greater
- Greater Than
- Is Not

Relational Operators Used on Date/Time Fields

The relational operators that can be used on date/time fields are listed below.

Comparing both date and time parts of date/time fields	• Before • On or Before • On • On or After • After • Not On
Comparing only the date part of date/time fields	• Before Date • On or Before Date • On Date • On or After Date • After Date
Matching all dates starting with the date that was the specified a number of days or weeks ago	• Last (n) Days • Last (n) Weeks
Matching all the dates prior to and including the date that was the specified number of days or weeks ago	• Older Than (n) Days • Older Than (n) Weeks

Note: In date fields, StarTeam treats blanks as zeroes. That means that “no date” is less than any specific date. For example, if you write a query that searches for change requests that were closed prior to some specific date, all the change requests with no date in the **Closed On** field are included in the results, even though they have not been closed yet. It is easy to eliminate the change requests that contain blanks in the **Closed On** field from such a query. You simply AND the condition that searches for change requests closed on or before a specific date with another condition that searches for change requests closed after the date zero.

Copying Queries

StarTeam allows you to create new queries quickly by copying an existing query and editing it. Using this feature saves time because you do not have to recreate the query conditions.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Select a query from the list.
5. Click **Copy**. The **Copy Query** dialog box opens.

   Tip: Public queries have a multi-user icon to the left of the query name. Private queries have a single-user icon.
6. Type a name for the query in the **Query Name** field.

7. Select the **Public** check box to add this query to the project (and the server configuration), allowing anyone with the appropriate access rights to use it. If you do not check the **Public** check box, the query will be private, that is, available only to your user ID.
8. Click **OK**. The new query displays in the **Queries** dialog box.
9. To change the conditions in the query, select it from the **Queries** dialog box and click **Edit**. The **Edit Query** dialog box opens.
10. Edit the appropriate nodes of the tree.
11. Click **Save**. The **Queries** dialog box opens.

    Note: If you do not have the access rights to create a public query for this project, you can create a private query.
12. Click **Close**.

Note: If this is a public query, you might want to set access rights for it.

Editing Queries

To display a useful set of data, you might need to edit or add to a query.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Select a query.
5. Click **Edit**. The **Edit Query** dialog box opens.
6. Edit the appropriate nodes of the tree.
7. Click **Save**. The **Queries** dialog box opens listing the edited query.
8. Click **Close**.

Deleting Queries

You can delete queries that you are sure you no longer use.

Note:

- You must have the appropriate access rights to delete a public query.
- You cannot delete a query that is referenced by a filter.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Filters** > **Queries** . The **Queries** dialog box opens.
4. Select a query.
5. Click **Delete**.
6. Click **OK**. The query is removed.
7. Click **Close**.

Predefined Queries

Component > **Filters** > **Queries**

Displays the **Queries** dialog box. Initially, the dialog box lists default queries that are predefined for the current component. You can apply a default query, edit a default query, create a new query, or delete a default query.

Note: Default queries that do not appear in your list might have been deleted after installation.

File	`Files to Check In`, `Files to Check Out`, `Flagged Items`.
Change Request	`Flagged Items`, `Not a Priority`, `Priority`, `Status=Closed`, `Status=Open`, `Status=Resolved`, `Status=Verified`, `Type=Defect`, `Type=Suggestion`, `Unread Changes`.
Requirement	`Flagged Items`, `I Am Responsible`.
Task	`Flagged Items`.
Topic	`Flagged Items`, `I Am Recipient`, `Show Active`.
Folder	`Folders Not In View`.
Audit	None.

Individual Query Access Rights

The following table describes the individual query access rights:

Generic item rights	
See object and its properties	See this query in the **Queries** dialog box and view its properties in the **Edit Query** dialog box.
Modify properties	Change the properties for this query. The properties that can be modified are its name and conditions.
Delete object	Delete this query from the list of queries.
Change object access rights	Change the access rights for this query.

Setting Access Rights for a Query

1. Click **Filters** > **Queries** from the appropriate component (file, change request, requirement, and so on). The **Queries** dialog box opens.
2. Select the query and click **Access Rights**. The **Query <Query Name> Access Rights** dialog box opens.
3. Click **Add**. The **Assign Access Rights To** dialog box opens.
4. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
5. Select **Grant** and click **OK**.

   Caution: Never select **Deny** unless you are creating an exception. Deny records must be created before grant records.
6. Select/clear the appropriate check boxes. Selecting or clearing the check box for a category, such as **Generic object rights** for a project, selects or clears all the access right check boxes for that category. The category check box has only two states. When it is cleared, the access right check boxes for that category are either all cleared or mixed: some selected and some cleared.

   Caution: Clicking **Delete** removes the selected user or group from the **User and Groups** list in the **Access Rights** dialog box. The selected user or group loses any previously set access rights to the Server.
7. Click **OK**.

Filters

A filter is a named arrangement of data that consists of a set of fields (used as column headers), sorting and grouping information, and (usually) a query. Once a filter has been created, it can be used in every project that has the same server configuration.

Filter names are not case sensitive. For example, if you have a filter named `recent CRs`, you cannot create a filter named `Recent CRs`, as StarTeam considers the two filters to be identical. In the **Filters** list, filters display in alphanumeric order, but you can control the order in which they appear by carefully naming or renaming them.

If you set up a filter and do a **Send to** in the client, only the fields displayed by the filter are sent to the recipient.

You can filter data in the upper pane in several different ways:

- By applying an existing filter.
- By arranging the data (changing displayed fields, sorting and grouping the files, and so on) and applying a query. You can then use this arrangement as the basis for a new filter.
- By creating a new filter from scratch.

Note: Only private queries can be used in private filters, and only public queries can be used in public filters. Therefore, you cannot copy a filter and change the status of the new filter unless the filter does not include a query.

Creating Filters

To limit the data shown on the upper pane, you can create a filter. Filters can be based on the current arrangement of data in the upper pane or created from scratch. Once a filter has been created, it can be used in any project in the same server configuration by any user with the appropriate access rights.

You can also create a new filter based on an existing filter by copying the existing filter.

Note: If you set up a filter and do a **Send to** in the client, only the fields displayed by the filter are sent to the recipient.

Creating a New Filter

1. Right-click a column header on upper pane and choose **Filters**. The **Filters** dialog box appears.
2. Click **New**. The **New Filter** dialog box opens.
3. Type a name for this filter in the **Filter Name** field.
4. Check **Public** if you want to add this filter to the project so that anyone with the appropriate access rights can use it instead of making it available to your user ID.
5. Click **OK**.
6. Click any of the following buttons in the **Filters** dialog box and specify the options:

Fields	Select the column header fields.
Sort, Group	Sort and group items in up to four fields in ascending or descending order.
Query	Limit the items that appear in the upper pane to those that match the query.

7. Click **Context**, for files only, and specify the files that will be affected by the filter.

   Clicking this button opens the **Set Filter Type** dialog box where you apply the filter to one of the following by selecting an option button:

Items in the view	Equivalent to applying both your filter and the **Files in view** filter.

Items not in the view Equivalent to applying both your filter and the **Files not in view** filter.

All items not excluded from the view Equivalent to applying both your filter and the **<All Non-Excluded Files>** filter.

8. Click **OK** to return to the **Filters** dialog box.
9. Click **OK** to apply the filter.

Note: If this is a public filter, you can set individual or component-level access rights for it.

Creating a New Filter from the Current Arrangement

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Sort and group the data shown on the upper pane, as desired.
4. Right-click a column header in the upper pane and choose **Save Current Settings** from the context menu. The **Save Current Settings** dialog box opens.
5. Type a name for this filter in the **Filter Name** field.
6. Do one of the following:
   - Check **Public** to add this filter to the project so anyone with the appropriate access rights can use it.
   - Uncheck **Public** to make the filter private, available only to your user ID.
7. Click **OK**.

Applying Predefined Filters

Existing public filters can be used on all projects in the same server configuration by any team members who have the appropriate access rights. Private filters can be used only by you.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Select a filter from the **Filters** list.
4. Click **OK**.

Editing Filters

You edit filters by changing their fields, sort orders, or queries.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Choose **Filters** > **Filters**. The **Filters** dialog box appears.
4. Select a filter from the **Filters** list.
5. Edit any of the following:

   Fields button Select the column header fields.

   Sort, Group button Sort and group items in up to four fields in ascending or descending order.

   Query button Limit the items that appear in the upper pane to those that match the query.

   Context button (for files only) Specify the files that will be affected by the filter. Clicking this button opens the **Set Filter Type** dialog. On this dialog box, apply the filter to one of the following by selecting an option button: **Items in the view** is equivalent to applying both your filter and the **Files in view** filter. **Items not in the view** is equivalent to applying both your filter and the **Files not in view** filter. **All items not excluded from the view** is equivalent to applying both your filter and the filter.

6. Click **Save As**. The **Save Filter As** dialog box appears. Do *not* change the name of the filter.
7. Click **OK** to return to the **Filters** dialog box.
8. Do one of the following:
   - Click **Select** to apply the edited filter to the upper pane.
   - Click **Close** to exit without applying the edited filter.

Copying Filters

To save time, you can create a new filter by basing it on an existing filter. But if the original filter includes a query, the new filter must have the same status (public or private) as the original filter. The reason for this is that only public queries can be used with public filters and only private queries can be used with private filters.

If you want to create a public filter based on a private filter with an associated private query, a work around exists.

Copying a Filter

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Choose **Filters** > **Filters**. The **Filters** dialog box appears.
4. Select a filter from the **Filters** list.
5. Click **Save As**. The **Save Filter As** dialog appears.
6. Type a name for this filter in the **Filter Name** field.
7. Select or clear the **Public** check box. If the filter includes a query, the status of the new filter must be the same as the status of the original filter.
8. Click **OK**.
9. Do one of the following:
   - Click **Select** to apply the new filter to the upper pane.
   - Click **Close** to exit without applying the new filter.

Copying a Private Filter and Changing its Status

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Do one of the following:
   - Right-click a column header on upper pane and choose **Filters**.
   - Choose **Filters** > **Filters** from the component or context menu.
4. From the list, select a private filter that has a query attached to it. For private filters, the **Public** check box is clear. The **Query** list shows the query associated with a particular filter.
5. Click the **Query** button. The **Query** dialog box appears, with the selected query highlighted.
6. Click the **Copy** button. The **Copy Query** dialog box appears.
7. Type a name for the new query, and select the **Public** check box to change the status of the query. Click **OK**. This action re-displays the **Queries** dialog box.
8. Highlight the new query, and click **Select**. The **Filters** dialog box appears.

9. Select the filter that has the new public query, and click **Save As**. Type a name for the new filter, and select the **Public** check box. Click **OK**. This action creates a new public filter with an attached public query, while the original private filter with its private query remains the same.

Changing a Private Filter to a Public Filter

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Do one of the following:
   - Right-click a column header on upper pane and choose **Filters**.
   - Choose **Filters** > **Filters** from the component or context menu.
4. From the list, select a private filter. For private filters, the **Public** check box is clear.
5. Select the **Public** checkbox.
6. Click **OK**. A message appears noting that associated queries will also become public.
7. Click **Yes**. The filter and associated queries are now public.

Deleting Filters

If desired, you can delete filters that you no longer use.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab.
3. Choose **Filters** > **Filters**. The **Filters** dialog box appears.
4. Select a filter from the **Filters** list.
5. Click **Delete**.
6. When a message box asks you to confirm your deletion, click **OK**. This action returns you to the **Filter** dialog box.
7. Click **Close**.

Filtering Process Tasks From Other Tasks

If you have previously enabled enhanced process links in your project so that StarTeam created process tasks, you can filter your tasks to separate the process tasks from the regular tasks. Use the **Usage** field value to determine the difference between process tasks and standard tasks. If the **Usage** value is anything other than **Other**, then it is a process task.

1. Choose **Task** > **Filters** > **Filters** . The **Filters** dialog box appears.
2. Click **New** and give the new filter a name.

   Alternatively, copy an existing filter by selecting it, clicking **Save As**, and giving it a new name. Then select the copied filter and continue with the next steps.
3. Click **Fields** in the **Filters** dialog box.
4. Move the **Usage** field from the **Available Fields** list to the **Show Fields in this Order** list and click **OK**.
5. Click **Query** to open the **Queries** dialog box, and click **New**. This opens the **Edit Query** dialog box.
6. Type a **Name** for the new query, and choose the following in the **Condition Node** section: **Field** = **Usage**, **Operator** = **Not Equal**, and **Value** = **Other**.
7. Click **Add** to add the condition to the query, and **Save** to save the query and return to the **Queries** dialog box. Your new query is now highlighted in the list of queries.
8. Click **Select** in the **Queries** dialog box use this query in your new filter. You are returned to the **Filters** dialog box, and your new filter should be highlighted.

9. Click **Save As** to save the filter.

   Notice your new filter has been added to the **Filter** list at the top of the client.

To use the filter, simply select it from the **Filter** list on the **Task** tab. Conversely, you can create a filter that displays only the standard tasks. In the query, use the condition **Usage Equals Other**.

Tip: If you use tasks on a regular basis, and not just for process tasks, add **Usage Equals Other** to existing queries so you never see process tasks when working on tasks that have been manually created, or imported from usingStarTeam Microsoft Project Integration.

Filtering StarTeam Agile Tasks From Other Tasks

If you are using StarTeam Agile with StarTeam, you can filter your tasks to separate the tasks created in StarTeam Agile from the tasks created in the StarTeam Cross-Platform Client. Use the **Type** field value to determine the difference between StarTeam Agile tasks and standard tasks. If the **Type** value is anything other than **Unknown**, then it is a StarTeam Agile task.

1. Choose **Task** > **Filters** > **Filters** . The **Filters** dialog box appears.
2. Click **New** and give the new filter a name.

   Alternatively, copy an existing filter by selecting it, clicking **Save As**, and giving it a new name. Then select the copied filter and continue with the next steps.
3. Click **Fields** in the **Filters** dialog box.
4. Move the **Type** field from the **Available Fields** list to the **Show Fields in this Order** list and click **OK**.
5. Click **Query** to open the **Queries** dialog box, and click **New**. This opens the **Edit Query** dialog box.
6. Type a **Name** for the new query, and choose the following in the **Condition Node** section: **Field** = **Usage**, **Operator** = **Not Equal**, and **Value** = **Other**.
7. Click **Add** to add the condition to the query, and **Save** to save the query and return to the **Queries** dialog box. Your new query is now highlighted in the list of queries.
8. Click **Select** in the **Queries** dialog box use this query in your new filter. You are returned to the **Filters** dialog box, and your new filter should be highlighted.
9. Click **Save As** to save the filter.

   Notice your new filter has been added to the **Filter** list at the top of the client.

To use the filter, simply select it from the **Filter** list on the **Task** tab. Conversely, you can create a filter that displays only the standard tasks. In the query, use the condition **Usage Equals Other**.

Resetting Filters

In StarTeam, you can apply a filter, then rearrange the data on the upper pane or apply a new query. Doing this places an asterisk in front of the filter's name, showing that it has been changed. After looking at the new data, you can then reset the filter as it was originally defined on the server, which removes the asterisk.

1. Choose **Filters** > **Filters**. The **Filters** dialog box appears.
2. Do one of the following:

   - Right-click a column header on the upper pane, then select **Reset Current Settings**.
   - Choose **Filters** > **Reset Current Settings** from the component or context menu.

   The system asks: `Reset filter: <Filter>?`
3. When the system asks: `Reset filter: <Filter>?`, click **OK**. This action resets the filter and removes the asterisk.

Sorting and Grouping Data

You can choose to do a primary sort in the upper pane (based on one column), or a more complicated sorts up to a fourth order.

Performing a Primary Sort on One Column

1. Open the view on the data you wish to sort or group.
2. Click a column header to sort the data in the upper pane based on the value in that column. The sort is in ascending order by number, letter, internal order, or internal key, depending on the data.
3. Click the column header again to reverse the sort order.

A triangle appears on the column header of the sorted column. The triangle points upward for ascending sorts and downward for descending sorts.

Note: You can also sort the data in the lower pane when the **Link** tab is selected.

Performing Up to a Fourth-Order Sort

1. Do one of the following:

   - Right-click a column header on upper pane and choose **Sort and Group**.
   - Right-click in the upper pane and choose **Filters** > **Sort and Group**.

   The **Sort and Group** dialog box displays four group boxes, each indented slightly more to the right than the one above it. The first group box designates a primary sort order, the second designates a secondary sort, and so on.
2. Optionally, check the **Show Advanced Fields** check box at the bottom of the dialog box to list all the fields in **First By** and **Then By** lists. Some fields are rarely used and considered advanced.
3. Select a field from the **First By** list. If you are grouping the items, the field does not need to be displayed in the upper pane. If you are not grouping the items, you can sort them based on a field that is not displayed, but you will not be able to tell where one group leaves off and the next begins.
4. Select the **Ascending** or **Descending** option button. The default setting is ascending order.
5. Select **Group By** to group the items which have the same values in this field. If you do not select any additional sort options, text fields are sorted in ASCII order. Enumerated and user ID fields are sorted by their internal order or internal keys. That is, enumerated fields are sorted in the order given to them by the person who created the field; user ID fields are sorted in the order in which they were created. The application disables the Sort Options button for numeric and date/time fields.
6. Optionally, click **Sort Options** for additional sorting selections. The **Sort Options** dialog box appears.

   - Select **As Text** to sort enumerated and user ID fields by the names of their possible values. For text fields, **As Text** is your only choice.
   - Uncheck the **Case-sensitive** check box to sort alphabetically or check it to sort in ASCII order (where uppercase letters precede lowercase letters).
7. Add secondary and lower order sorts by using the **Then By Group** boxes as needed.

File Filters

File > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<All Files By Status>	Groups the files that have the same status: **Current**, **Deleted on Disk**, **Deleted on Server**, **Missing**, **Modified**, **Modified on Disk and Deleted on Server**, **Merge**, **Out Of Date**, **Not In View**, and **Unknown**.
<Flagged Items>	Files that have been flagged for some special reason. For example, you may want to use flags to remind yourself to follow up on a customer request. Flags are set, viewed, and removed by the user who created them.
All Non-Excluded Files	All non-excluded files that exist either in application folders or their working folders.
Files In View	Files in the working folder that exist in the current project view.
Files Not In View	Files in the working folder that do not exist in the current project view. Unless you add them to the project, their names will never appear on the same list as the files that are in your project.
Files to Check In	All files in the view that need to be checked in. The statuses are **Modified**, **Merge**, or **Not In View**.
Files to Check Out	All files in the view that need to be checked out. The statuses are **Out Of Date**, **Missing**, or **Merge**.

Note: StarTeam lists the files that need to be merged when you apply either the **Files To Check In** or **Files To Check Out** filter.

Change Request Filters

Change Request > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<Show All>	Displays all items.
By Status and Responsibility	Groups change requests based on their statuses and the users who are currently responsible for processing the requests.
Not a Priority	Displays only the change requests that are not a priority.
Priority	Displays only the change requests that are a priority.
Show Unread Changes	Displays only the change requests that you have not read (or not read since they were modified).
Status = Closed	Displays only the change requests that are closed.
Status = Deferred	Displays only the change requests that are postponed.
Status = Open	Displays only the change requests that are open and in progress.
Status = Resolved	Displays all the change requests that have one of the following statuses: **As Designed**, **Cannot Reproduce**, **Documented**, **Fixed**, or **Is Duplicate**.

Status = Verified	Displays all the change requests that have one of the following statuses: **Verified As Designed**, **Verified Cannot Reproduce**, **Verified Documented**, **Verified Fixed**, or **Verified Is Duplicate**.
Type = Defect	Displays only the change requests that have the type **Defect**.
Type = Suggestion	Displays only the change requests that have the type **Suggestion**.

Requirement Filters

Requirement > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<Show All>	Displays all items.
<Flagged Items>	Lists only requirements that have been flagged.
Grouped by Creator	Displays groups of requirements, one group for each user who has created requirements.
Grouped by Status	Displays groups of requirements, one group for each existing status.
I Am Responsible	Displays only the requirements for which you are responsible.

Folder Filters

Folder > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<Show All>	Displays all items.
Folders Not In View	Displays only folders that are not in the current view.

Task Filters

Task > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

Examples of custom task filters that you might create include:

- `Responsibility Equals <user name>`, which identifies only the tasks for which a specific person is responsible.
- `Percent Complete < 100`, which identifies unfinished tasks.

<Show All>	Displays all items.

Topic Filters

Topic > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<I Am Recipient>	Identifies all the topics that name you as a recipient.
By Creator	Groups the topics by their original authors.
Show Active	Identifies all topics and responses that have `Active` status.
<Show All>	Displays all items.

Audit Filters

Audit > **Filters** > **Filters**

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of filters depends on your selection from the folder tree and whether the **All Descendants** button is selected from the toolbar or component menu.

StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

<By Class and Event>	Displays audit entries sorted by their value in the **Class Name 1** field (type of item) and **Event** (type of action) field.
By Transaction and Event	Groups audit log entries by descending **Transaction ID** and then by **Event** type. This filter provides a reverse-chronological list of updates in the view by transaction.
Events	Groups audit log entries by **Event** type, then by **Target 1 Class ID**, and then by **Created Time**.
Show All	Displays all entries (the default).

Note: You can limit the number of audit log entries displayed by creating a query that selects audit log entries by specific property values.

View Compare/Merge Session Filters

Filtering allows you to limit the types and numbers of items that appear in the upper pane. The list of available filters depends on the perspective you are using in the View Compare/Merge session. StarTeam provides a set of predefined filters that are intended as starting points for you to create your own custom filters. Use the **Filter** list on the toolbar to view and apply predefined file filters.

You can create additional filters for the **Merge Perspective** and the **Test Perspective**. To access the **Filters** dialog box, right-click a column header in the upper pane of the **Merge Perspective** or **Test Perspective** and choose **Filters**. The **Compare Perspective** has two predefined filter lists, but you cannot create any additional filters for that perspective.

<Show Type>	Displays only items of the selected type in the **Compare Perspective**.

<Show Items With Differences>	Displays items from both the source and target views in the **Compare Perspective** that have differences.
<Show Unresolved Items>	Displays only items in the **Compare Perspective** with a merge status of **Unresolved**.
<Show All Items>	Displays all items of the selected type in the **Compare Perspective**.
Show Items: [action]	Displays only items in the **Compare Perspective** with the selected merge action.
<VCM: Items By Merge Status>	Displays items in the upper pane of the **Merge Perspective** sorted by merge status. This filter name is dynamic, depending on which component tab is selected. You can display items by merge status for files, change requests, and folders.
<VCM: Merge Properties>	Displays the items in the upper pane of the **Merge Perspective** with the primary descriptor in column one, and all the remaining VCM properties in the following columns.
Merge Action	Sorts items in the upper pane of the **Merge Perspective** and **Test Perspective** the by merge action.
<Show Items With Updates>	Displays only items from the source view in the **Compare Perspective** that will result in updates to the target.

Note: In the **Test Perspective**, you also see the filters for files when the **Files** tab is selected, for folders when the **Folders** tab is selected, and so on.

Individual Filter Access Rights

The individual filter access rights are described below:

Generic item rights

See object and its properties	See the filter in the filters list (on the toolbar) and view its properties in the **Filters** dialog box.
Modify properties	Change the properties for the filter. The properties that can be modified for the filter are its list of displayed fields, its sorting and grouping rules, the query associated with it, and its context (the items of the component to which it can be applied).
Delete object	Delete the filter from the list of filters
Change object access rights	Change the access rights for the filter.

Setting Access Rights for a Filter

1. Choose **Filters** > **Filters** from the appropriate component (file, change request, requirement, and so on). The **Filters** dialog box opens.
2. Select the filter, and click **Access Rights**. The **Filter <Filter Name> Access Rights** dialog box opens.
3. Click **Add**. The **Assign Access Rights To** dialog box opens.
4. Select a user or group. Users are listed by their user names and groups are listed by their paths (excluding the `All Users` group).
5. Select **Grant** and click **OK**.

Caution: Never select **Deny** unless you are creating an exception. Deny records must be created before grant records.

6. Select/clear the appropriate check boxes. Selecting or clearing the check box for a category, such as **Generic object rights** for a project, selects or clears all the access right check boxes for that category. The category check box has only two states. When it is cleared, the access right check boxes for that category are either all cleared or mixed: some selected and some cleared.

   Caution: Clicking **Delete** removes the selected user or group from the **User and Groups** list in the **Access Rights** dialog box. The selected user or group loses any previously set access rights to the Server.
7. Click **OK**.

Exporting

This section provides information about exporting data.

Exporting Data

You are limited to exporting 60 columns of data that display in the upper pane. Databases have limitations as well. Verify that the application in which you intend to use the exported data can accept all 60 columns. Exported data can be imported into Microsoft Access, Microsoft Excel, or other applications.

To export data that can be imported into other applications follow the steps below.

1. Select a folder from the folder tree or in the upper pane on the **Folder** tab.
2. Click a component tab. If you are exporting specific items in the upper pane, also select the specific items.
3. Right-click in the upper pane and choose **Advanced** > **Export** from the context menu.

   Note: For the **Audit** component only, select **Export** from the context menu.

   This opens the **Export** dialog box.
4. Move the fields you want to export from the **Available fields** list to the **Show these fields in this order** list.

   Move them in the order you want them presented in the target application.
5. Specify which items to export in the **Export for** group.

   You can export selected items or all items in the upper pane.
6. Select the desired export options in the **Options** group.
7. Type a path and filename in the **Output file name** field, or browse to an existing file to overwrite it.

   Note: Use `.txt` as the filename extension, or another extension which the target application can import. By default, the file is assigned a `.txt` extension.
8. Click **OK**. The export file is generated.
9. Open the target application and import the exported file.

   Print a report if desired.

Export Dialog Box Options

To specify export options, open the **Export** dialog box. Right-click in the upper pane and choose **Advanced** > **Export** .

Use the **Export** dialog box to export date in the upper pane to a text-delimited file for import into other applications, such as spreadsheets and database applications. Below are the fields and options in the **Export** dialog box.

Available Fields Lists all the fields displayed as columns in the upper pane which are available for export. You are limited to displaying 60 columns in the upper pane, and therefore, are limited to exporting 60 columns of data. The target application must be able to import the number of fields you are exporting.

Advanced Fields Displays the advanced fields in the **Available Fields** list.

Show These Fields In This Order Lists the fields you selected for export. The fields will be exported in the order they appear in this list.

Current Selection Specifies that the export should only include the items currently selected in the upper pane on the active component tab .

All Items Specifies that the export should include all the items in the upper pane on the active component tab.

Export Field Names Includes the column header fields for the report.

Replace Embedded Carriage Returns With Spaces Exports the data without embedded carriage returns/line feeds.

Text Qualifier Specifies the text qualifier to use at the beginning and end of text fields for preventing the special treatment of carriage returns/line feeds and other special characters that occur in HTML output.

Delimiter Specifies the character to use to separate the data fields in the exported text file. You may enter a custom delimiter or choose from the predefined list.

Encoding Specifies the encoding to use for the exported data.

Output File Name Specifies the location and name of the output text file. The file must have a `.txt` extension, or another extension the target application will import. The default extension is `.txt`.

Reports

The StarTeam Cross-Platform Client has two types of reports available:

***Rich* BIRT-based reports** These reports allow you to access the BIRT report designer to create your report.

***Classic* HTML reports** These reports are based on StarTeam Cross-Platform Client-specific configuration files that you use out of the box or modify for your needs.

Both report types are accessed using the **Reports** dialog box.

Rich Reports

The StarTeam Cross-Platform Client offers Rich reports based on the BIRT report designer to create custom reports. You can easily select your data source and tables and fields from the StarTeam schema to customize your report.

For more information about BIRT reports and the BIRT report designer, refer to *http://eclipse.org/birt/*.

Important: Rich reports follow the same naming convention as the classic reports. When you are finished creating your report, move the `.rptdesign` file created with BIRT into the same folder as the classic reports so that it is available to run within the StarTeam Cross-Platform Client.

Tip: To customize the directory where the reports are located, open and edit the `starteamcp.stjava{32|64}` file. Add the following option:

```
-Dstarteam.reports.dir=C:\PathToReports
```

Creating a New Report Structure

Use the following steps to create a rich report in the StarTeam Cross-Platform Client .

1. Open the StarTeam **Reports** dialog box by right clicking an item and selecting **Reports**. The **Reports** dialog box opens.
2. Select the **Rich Reports** option.
3. Enter in a name in the **Output file name** field.
4. Enter a title in the **Report** title field.
5. To configure the report, click **Open Report Editor**. The **BIRT Report Designer** opens.
6. Click **File** > **New** > **Report**. The **New Report** dialog box opens.
7. Select the folder for the report and enter the **File name**.
8. Click **Next**.
9. On the **Report** screen, select the type of report to create from the **Report templates** list.
10. Click **Finish**.

Create the Data Source

1. From the **BIRT Report Designer**, click **Data** > **New Data Source**.
2. On the **New Data Source** page, select **StarTeam Data Source**.
3. In the **Data Source Name** field, enter the data source name. For example: `localhost`.
4. Click **Next**. The **New StarTeam Data Source Profile** page open.
5. Enter the following fields for the data source:

Server Address	Address of the StarTeam Server. For example: `localhost`.
TCP/IP Endpoint	Port of the server.
Project	The name of the StarTeam project to use.
View	The name of the StarTeam view to use that exists in the project specified above.
User name	The StarTeam user to log in as.
Password	The user's password.

6. Click **Finish**.

Create the Data Set

1. From the **BIRT Report Designer**, click **Data** > **New Data Set**. The **New Data Set** page opens.
2. Select an existing **Data Source**.
3. Select `StarTeam Data Set` as the **Data Set Type**.
4. Enter a name in the **Data Set Name** field.
5. Click **Next**. The **Select Query** page opens.
6. Use the **Select Query** page to model commands that you would use in the command line. Restrictions on this page are based on how some types will or will not work with other queries or types.

   The **Query Text** field is editable and allows you to create your own (advanced) query. For example, you can build a `where` clause with property names: `select * from File where propertyNameA`

`="A" and propertyNameB="Bla"`. You can also restrict the columns returned by saying `select propertyNameA, propertyNameB from file`.

If the server is down when you open the **Data Set** page, an error is displayed and all fields will be disabled except for **Query Text**. This is because many fields retrieve from the server. You can still edit/ save the **Query Text**.

If you share a report with another user, the **DataSource** credentials will only decrypt if you have the same user name. If you try to edit the data set or data source, an error will be reported telling you that decryption failed and to re-enter the credentials.

7. Click **Check** when you are satisfied with your query. You will be notified of the results.
8. Click **Finish**. The **Output Columns** page opens.
9. Use the **Output Columns** page to change display names or change types. The **Edit** button allows you to change the default fields for the data set. For example, you could change the display name of a column or change a *string* data type to an *Integer* data type, as long as the format is correct.
10. Click **OK**. The **Preview Results** page opens with your results.
11. Click **OK**. BIRT will display all columns in the tree allowing you to drag them onto the report, create charts, or use any other BIRT functionality that you need.

Classic Reports

StarTeam offers a wide variety of *Classic* reports that are pre-formatted in HTML. However, you can modify the HTML formatting or the columns used in the report by modifying the correct template for the report. You can also use StarTeam Datamart to extract data from StarTeam Server and place the data into a relational database where reporting tools (Crystal Reports and Business Objects) can access it.

Report generation is affected by sorting, grouping, and selecting items in the file, change request, topic, or task list of the view window. Before you generate a report, arrange the data in the upper pane of the client.

This topic provides some examples of why you might generate reports and information about customizing report templates. Reports are generated for a variety of reasons. The following examples describe three possible scenarios for generating a report using change requests and a brief walkthrough on how to generate them.

Although these examples apply only to change requests, you can create additional reports for other items. You can also export data for use in spreadsheets and other applications.

Manager wants a report of all CRs resolved across a project

1. The manager selects the root folder from the folder hierarchy, and selects the **Change Request** tab. The list of change requests display in the upper pane.
2. The manager selects the `Status=Resolved` filter from the **Status** list.
3. The **All Descendants** (either from the toolbar or the **Change Request** menu) button is activated.
4. Finally, he or she selects **Change Request** > **Reports** to generate a **Change Request Summary Report**.

Team leader wants to see CRs based on responsibility

1. The team leader selects the project folder that corresponds to his or her team from the folder hierarchy, and selects the **Change Request** tab. The list of change requests display in the upper pane.
2. The team leader selects the `<Show All>` filter from the **Status** list.
3. The **All Descendants** button is activated.
4. The team leader clicks the header of the **Responsibility** column to sort the change requests based on the responsible team member.
5. Finally, the team leader selects **Change Request** > **Reports** from the main menu to generate a **Change Request Detail Report**.

Developers want to see CRs for which they are responsible

1. The individual programmers select their project folders from the folder hierarchy, and select the **Change Request** tab. The list of change requests display in the upper pane.
2. The programmer applies a query (`Responsibility Equals <username>`) to view only his or her assigned change requests.
3. The **All Descendants** button is activated.
4. Finally, the programmer selects **Change Request** > **Reports** to generate a **Change Request Detail Report**.

Report Templates

StarTeam allows you to customize report templates. The templates are located in the folder you designated during installation. For example, if you used the default installation path for the Cross-Platform Client on a Windows platform, the `Reports` folder is `C:\Program Files\Micro Focus\StarTeam Cross-Platform Client <version>\Reports`. Be aware that different clients and different releases of just one client will probably have different installation folders. You may need to put templates in more than one location. All the templates are in HTML format.

You can open and edit the report templates in any text editor, Microsoft's Developer Studio, or HTML tool. A simple and easy method of creating and editing templates is to use Microsoft Word, which includes automated HTML file generation.

The `Reports` folder includes a series of templates for each type of report. Each template provides the formatting information needed to create a part of the report. For example, the **Change Request Default** report uses the following templates:

- `ChangeDefault.Title`
- `ChangeDefault.GrpInfo`
- `ChangeDefault.Group1`
- `ChangeDefault.EndReport`

`xDefault.Title` Processed first and only once. It uses the title you provide in the **Reports** dialog as the **Report Title**.

`xDefault.GrpInfo` Processed once for each group. Although you can sort data by clicking column headers, this does not result in groups for a report. You must use the **Sort and Group** feature to arrange groups.

`xDefault.Group1` Repeated for each item, in this case, each change request, in the report. It creates a record for the item and separates it from the records for other items with a horizontal line.

`xDefault.EndReport` Processed only once. It totals the number of items in the report.

Creating *Classic* Reports

All *Classic* reports that you create in the application show all or some portion of the data displayed in the upper pane. All *Classic* reports are generated in `HTML` format.

Note: You can also create reports using StarTeam Datamart. Refer to the *StarTeam Datamart User Guide* for more information.

1. Select a folder from the folder tree.
2. Click a component tab.
3. Choose **Reports** from the component or context menu. The **Reports** dialog displays the **Available Reports** list.
4. Select the ***Classic* Reports** option.

5. From the **Available Reports** list, choose the type of the report you want to generate.
6. Do one of the following:
   - To include only the items selected on the upper pane, select the **Current Selection** option button.
   - To include all items displayed in the upper pane, select the **Select All** option button.
7. Type or browse to the path and report filename in the **Output file name** field.

   Be sure to use `.htm` or `.html` as the file extension.

   Note: By default, the default report filename uses the convention `<STReport><date><alphanumeric code>.html` (for example, `STReport2006-07-24T22-03-59Z.html`).
8. Type a name for your report in the **Report title** field.
9. Click **Generate** to view the report on screen. Your web browser opens and displays your report. In addition, the report is saved in the location specified above.

Customizing Fields in *Classic* Report Templates

You can change what fields appear in a *Classic* report by changing the fields specified in the report's template. Fields must be specified in the templates using their internal identifier. This is usually different from their display name (which is used as the column header, the name displayed in a pane, and the name in the report).

Field names in the report template files are delimited by the characters "~~". For example, `~~ReportTitle~~` appears in all of the `x.Title` report templates. In the Cross-Platform Client, whatever you type as the name of the report in the **Reports** dialog box becomes the title of the report.

Tip: To see all field display names, right-click the column header in the upper pane and choose **Show Fields** from the context menu.

1. Open the report template file in your favorite text editor.
2. Edit the file following standard `HTML` file conventions.
3. Save the file.

   Note: If you want to create a new report template that has this new field, save the file with a new filename. All related report template files should be created with the same root filename.

For example, the `TopicsSummary.Group1` template contains the following lines:

```
<TD WIDTH=450><FONT SIZE=”2”>~~Title~~</FONT></TD><TD WIDTH=0></TD>
<TD WIDTH=100><FONT SIZE=”2”>~~CreateUserID~~</FONT></TD><TD WIDTH=0></TD>
<TD WIDTH=120><FONT SIZE=”2”>~~CreatedTime~~</FONT></TD><TD WIDTH=0></TD>
```

If you wanted to add the `Read Status` field to this report, you would need to know that its internal identifier is `ReadStatus`. Then you might add the following line to the template:

```
<TD WIDTH=120><FONT SIZE=”2”>~~ReadStatus~~</FONT></TD><TD WIDTH=0></TD>
```

Remember that, even if you specify a field in a template, that field is skipped if it does not also appear as a column in the project view window's upper pane. You might want to create filters for specific reports prior to creating the report.

Customizing *Classic* Report Templates

You can customize *Classic* report templates. They are located in the folder you designated during installation. If you used the default path during installation, the `Reports` folder is `c:\Program Files\Micro Focus\StarTeam Cross-Platform Client <version>\samples\details-templates\`. Note that different clients and client installations on different operating systems will probably have different installation folders. You may need to put templates in more than one location. All the templates are in `HTML` format.

While creating custom templates, keep these things in mind:

- A report typically consists of multiple template files, each using the naming convention `<ComponentReportname.Purpose>`. For example, the Default report provided for the Change Request component has five template files: `ChangeDefault.Name`, `ChangeDefault.Title`, `ChangeDefault.GrpInfo`, `ChangeDefault.Group1`, and `ChangeDefault.EndReport`.
- A double tilde (~~) precedes and follows field names in reports. For example, in the `ChangeDefault.Title` and other “Title” template files, the field name `~~ReportTitle~~` is used.

1. From the `Reports` folder, copy all report template files with the same root name to the `Reports` folder using a different root name.

   For example, if the `Default` report for the file component is the most similar to the new report you want to create, copy all `FileDefault.x` template files using a name like `FileMYREPORT.x`. In this case, you would create new files named `FileMYREPORT.Name`, `FileMYREPORT.Title`, and so on. In the Cross-Platform Client, the new report would appear in the **Report** dialog box with the name `MYREPORT`, along with the original `Default` report.
2. Open and edit the new report template files in any text editor or HTML editor.

   Tip: A simple and easy method of creating and editing templates is to use Microsoft Word 97 or later, which includes automated HTML file generation.
3. Save the report templates you edited.

Printing *Classic*Reports

1. Create the report and display it in the browser.
2. Right-click in the browser and choose **Print**. The **Print** dialog box opens.
3. Set the print options and click **Print**.

Configuring the Output Path for *Classic* Reports

You can customize the location in which you store the*Classic* reports you create. If desired, when you create a report, you can select an alternative location.

1. Choose **Tools** > **Personal Options** to open the **Personal Options** dialog box.
2. Select the **Workspace** tab.
3. Type or browse for a path for the **Report output path** field. This path becomes the default location for all reports that you create using the application.

   Note: The path you specify for **Output filename** in the **Reports** dialog overrides the default report output path specified in **Personal Options**.
4. Click **OK**.

Templates

StarTeam provides many *Classic* reports for each component that you can use as is or customize. Each report is generated using a series of template files that reside in the folder you designated during installation. For example, if you used the default installation path for the Cross-Platform Client on a Microsoft Windows platform, the `Reports` folder is `c:\Program Files\Micro Focus\StarTeam Cross-Platform Client <version>\samples\details-templates\`.

To customize the reports, you can open and edit the report templates in any text editor or HTML tool.

Tip: To customize the directory where the reports are located, open and edit the `starteamcp.stjava{32|64}` file. Add the following option:

```
-Dstarteam.reports.dir=C:\PathToReports
```

Template Files

The template files provided for each report are listed below.

Default

```
<component>Default.Title
<component>Default.Name
<component>Default.GrpInfo
<component>Default.Group1
<component>Default.EndReport
```

Detail

```
<component>Detail.Title
<component>Detail.Name
<component>Detail.GrpInfo
<component>Detail.Group1
<component>Detail.EndReport
```

Detail with Description

```
<component>Detail with Description.Title
<component>Detail with Description.Name
<component>Detail with Description.GrpInfo
<component>Detail with Description.Group1
<component>Detail with Description.Group2
<component>Detail with Description.EndReport
```

Grouping Summary

```
<component>Grouping Summary.Title
<component>Grouping Summary.Name
<component>Grouping Summary.GroupSummary
<component>Grouping Summary.EndReport
```

History

```
<component>History.Title
<component>History.Name
<component>History.GrpInfo
<component>History.Group1
<component>History.Group2
<component>History.EndReport
```

Links

```
<component>History.Title
<component>Links.Name
<component>Links.GrpInfo
<component>Links.Group1
<component>Links.Group2
<component>Links.EndReport
```

Summary

```
<component>Summary.Title
<component>Summary.Name
<component>Summary.GrpInfo
<component>Summary.Group1
<component>Summary.EndReport
```

Summary with Description

```
<component>Summary with description.Title
<component>Summary with description.Name
<component>Summary with description.GrpInfo
<component>Summary with description.Group1
<component>Summary with description.EndReport
```

Understanding the Template Files

The report template file names are composed as `<component><name>.<type>`. The table below describes each template file type.

.Title In the `.Title` templates, the report title is represented with the `~~ReportTitle~~` tag. In these templates, you might want to add your company’s logo, change the font or size, etc.

.GrpInfo The `.GrpInfo` templates contain information describing the group, which is represented as the `~~GroupingInfo~~` tag.

.Group1..9 The `.Groupx` files are processed in the numerical order in which they are found. The numbers do not have to be contiguous. The application searches for `.Group1` through `.Group9` as it creates a record for each item in the report. The names of the fields in the component are individually specified within the double tildes (~~). You must use the correct SQL name. The fields’ data will replace the SQL name and the double tildes. To appear in the report, all the specified fields must be displayed in the upper pane at the time the report is generated. Otherwise, those fields are skipped.

.GroupSummary The `.GroupSummary` templates are used when totalling the items in a group. The group is represented with the ~~GroupingInfo~~ tag.

.EndReport This template ends your report. It may include the following total tags: `~~TotalHistoryCount~~` `~~TotalRecordCount~~` `~~TotalLinkCount~~`

Classic Reports for the Audit Component

Audit > **Reports**

You can view and print a number of different reports for this asset

Default Lists information located in the **Detail** pane, using one line for each field.

Grouping Summary Indicates the number of assets in each group plus the total number of assets.

Summary Lists each selected audit entry (or all of them when none are selected). The report gives the **Event** , **Created Time**, **User**, **Class 1**, **Item 1**, **Class 2**, **Item 2**,**Class 3**, and **Item 3** fields and the total number of selected audit entries.

Classic Reports for the Change Request Component

Change Request > **Reports**

You can view and print a number of different reports for this asset

Default Lists information located in the **Detail** pane, using one line for each field.

Detail Lists specific change request fields: **Modified Time**, **Responsibility**, **Entered On**, **Entered By**, **Last Build Tested**, **Status**, **Severity**, **Addressed In**, **Addressed By**, **Priority**, **Test Command**, **Type**, **Synopsis**, **Description**, **Work Around**, and **Fix**.

Grouping Summary Indicates the number of assets in each group plus the total number of assets.

History Lists the change request fields found in a **Detail** report followed by information about each revision of the change request: the revision number, its date and time, author, view, comment, and branch revision.

Links Lists the selected assets and any items linked to them.

Summary Lists each selected change request (or all of them when none are selected). The report gives the **Modified Time**, **Responsibility**, **Entered On**, **Entered By**, **Priority**, **Type**, **Status**, **Severity**, and **Synopsis** fields and the total number of selected change requests.

Classic Reports for the File Component

File > **Reports**

You can view and print a number of different reports for this asset

Default	Lists information located in the **Detail** pane, using one line for each field.
Detail	Lists files and their revision histories.
Detail with Description	Lists files, their descriptions and revision histories.
Grouping Summary	Indicates the number of assets in each group plus the total number of assets.
Links	Lists the selected assets and any items linked to them.
Summary	Lists each selected file (or all of them when none are selected). The report gives the **Name**, **Status**, **Locked By**, and **Revision** fields and the total number of selected files.
Summary with Description	Lists each selected file (or all of them when none are selected). The report gives the **Name**, **Status**, **Locked By**, **Revision**, and **Description** fields and the total number of selected files.

Classic Reports for the Folder Component

Folder > **Reports**

You can view and print a number of different reports for this asset

Default	Lists information located in the **Detail** pane, using one line for each field.
Detail	Lists folders and their revision histories.
Detail with Description	Lists folders, their descriptions and revision histories.
Grouping Summary	Indicates the number of assets in each group plus the total number of assets.
Links	Lists the selected assets and any items linked to them.
Summary	Lists each selected file (or all of them when none are selected). The report gives the **File** , **Status**, **Locked By**, and **Revision** fields and the total number of selected folders.
Summary with Description	Lists each selected folder (or all of them when none are selected). The report gives the **File** , **Status**, **Locked By**, **Revision**, and **Description** fields and the total number of selected folders.

Classic Reports for the Requirement Component

Requirement > **Reports**

You can view and print a number of different reports for this asset

Default	Lists information located in the **Detail** pane, using one line for each field.
Detail	Lists specific requirement fields: **Modified Time**, **Name**, **Type**, **Status**, **Owner**, **Priority**, and **Description**.
Grouping Summary	Indicates the number of assets in each group plus the total number of assets.

History	Lists the change request fields found in a **Detail** report followed by information about each revision of the change request: the revision number, its date and time, author, view, comment, and branch revision.
Links	Lists the selected assets and any items linked to them.
Summary	Lists each selected requirement (or all of them when none are selected). The report gives the **Title**, **Created By**, and **Created Time** fields and the total number of selected requirements.

Classic Reports for the Task Component

Task > **Reports**

You can view and print a number of different reports for this asset

Detail	Lists specific Task fields: **Modified Time**, **Responsibility**, **Created Time**, **Created By**, **Name**, **Status**, **Priority**, **Milestone**, **Duration**, **Percent Complete**, **Needs Attention**, **Attention Notes**, **Planned Start**, **Planned Finish**, **Planned Work**, **Actual Start**, **Actual Finish**, **Actual Work**, and **Notes**.
Grouping Summary	Indicates the number of assets in each group plus the total number of assets.
Links	Lists the selected assets and any items linked to them.
Summary	Lists each selected task (or all of them when none are selected). The report gives the **Task Name**, **Created By**, and **Created Time** fields and the total number of selected tasks.

Classic Reports for the Topic Component

Topic > **Reports**

You can view and print a number of different reports for this asset

Default	Lists information located in the **Detail** pane, using one line for each field.
Detail	Lists specific Task fields: **Created By**, **Status**, **Created Time**, **Priority**, **Title**, and **Description**.
Grouping Summary	Indicates the number of assets in each group plus the total number of assets.
Links	Lists the selected assets and any items linked to them.
Summary	Lists each selected topic (or all of them when none are selected). The report gives the **Title**, **Created By**, and **Created Time** fields and the total number of selected topics.

Charts

StarTeam offers a wide variety of charts. The Cross-Platform Client allows you to create simple, distribution, correlation and time-series charts of your data. The charts you can create depend upon the component tab menu option that you have selected. For example, the charts available for files differ from those available for change requests.

Charts are created from the data displayed (maximum of 60 fields) in the upper pane. To select the data to be used for a chart, you can show or hide all descendants of a folder, sort and group items, and run queries and filters.

You can use charts in a number of different ways. For example, you can use charts to track the number of closed and newly-opened change requests during a time period of a product development cycle.

You can also filter out data in the upper pane of the Cross-Platform Client to display only the data that you want to include in your chart. You can select specific items from the filtered data to include in your chart.

Choosing the Chart Type

Once you have generated a chart, the **Chart** window opens displaying a default chart type. You can choose a different chart type.

1. Generate your chart.
2. Choose a different chart type from the list on the toolbar in the **Chart** window.

Chart Types

Area Emphasize the amount of change over a period of time, or they compare multiple items. An area chart also shows the relationship of parts to a whole by displaying the total of the plotted values. An area chart is a form of line chart, but the area between the horizontal (x) axis and the line connecting the data makers is filled with color. This makes it easy to see where the points encompassed by the different data series overlap.

Bar Show the changes in a data series over time. or they compare multiple items. Types of items are arranged vertically and data values are plotted horizontally to emphasize variation over time. The 3-D bar chart provides an extra dimension for plotting data by comparing values along two axes.

Bubble Are a type of scatter chart. The x and y coordinates of the data marker (the bubble) are determined by two data values. The size of the data marker indicates the value of a third variable.

Column Show the changes in a data series over time, or they compare multiple items. Types of items are arranged horizontally and data values are plotted vertically to emphasize variation over time. The 3-D column chart provides an extra dimension for plotting data by comparing values along two axes.

Heat Map Show the relationship between data items by using gradually changing shades of color. Heat map charts are commonly used in financial analysis to show which stocks are rising, which are falling and the amount and rate of change between them.

Line Emphasize the amount of change over a period of time, or they compare multiple items. Data points are plotted in series using evenly-spaced intervals and connected with a line to emphasize the relationships between the points.

Pie Show the size of items that make up a data series proportional to the total of the items in the series. Pie charts always show a single data series, and it is useful for determining which items in the series are most significant.

Scatter Used either to show the relationship of items in several distinct series of data, or to plot two sets of values as one series of x/y coordinates. Scatter chart draw attention to uneven intervals or clusters of data. This type of chart is often used to plot scientific data, and can highlight the deviation of collected data from predicted results.

Stack Bar Show the relationship of individual items in a series to the whole.

Stack Column Show the relationship of individual items in a series to the whole.

File Chart Fields

This table lists the available fields for file charts.

Simple	Distribution	Correlation	Time-series
Content Revision	Create charts based on fields currently being grouped.	Content Revision	Configuration Time
DotNotation ID	—	DotNotation ID	Created Time
EOL Character	—	EOL Character	Deleted Time
Object ID	—	Object ID	File Time Stamp at Check-in
Parent ID	—	Parent ID	Modified Time
Parent Branch Revision	—	Parent Branch Revision	Sync Local Time Stamp
Parent Revision	—	Parent Revision	Working File Time Stamp
Project ID	—	Project ID	—
Revision	—	Revision	—
Revision on Disk	—	Revision on Disk	—
Root Object ID	—	Root Object ID	—
Size	—	Size	—
Sync Branch Version	—	Sync Branch Version	—
Sync Content Version	—	Sync Content Version	—
Sync Local Size	—	Sync Local Size	—
Sync on Path to Root	—	Sync on Path to Root	—
Vault Branch Version	—	Vault Branch Version	—
Version	—	Version	—
Working File Size	—	Working File Size	—

Change Request Chart Fields

This table lists the available fields for change request charts.

Simple	Distribution	Correlation	Time-series
Attachment Count	Create charts based on fields currently being grouped.	Attachment Count	Closed On
CR Number	—	CR Number	Configuration Time
DotNotation ID	—	DotNotation ID	Created Time
Object ID	Object ID	Object ID	Deleted Time
Parent ID	—	Parent ID	Entered On
Parent Branch Revision	—	Parent Branch Revision	Modified Time
Parent Revision	—	Parent Revision	Resolved On
Root Object ID	—	Root Object ID	Total Open
Version	—	Version	Verified On

Requirement Chart Fields

This table lists the available fields for requirement charts.

Simple	Distribution	Correlation	Time-series
Ambiguities Found	Create charts based on fields currently being grouped.	Ambiguities Found	Created Time
Attachment Count	—	Attachment Count	Configuration Time
Children Count	—	Children Count	Deleted Time
Comment ID	—	Comment ID	Modified Time
Expected Effort	—	Expected Effort	End Modified Time
High Effort	—	High Effort	—
Low Effort	—	Low Effort	—
Number	—	Number	—
Object ID	—	Object ID	—
Parent	—	Parent	—
Requirement ID	—	Requirement ID	—
Responsible Count	—	Responsible Count	—
Revision Flags	—	Revision Flags	—
Version	—	Version	—

Task Chart Fields

This table lists the available fields for task charts.

Simple	Distribution	Correlation	Time-series
Actual Hours	Create charts based on fields currently being grouped.	Actual Hours	Actual Finish
Attachment Count	—	Attachment Count	Actual Start
Children Count	—	Children Count	Configuration Time
Estimated Hours	—	Estimated Hours	Constraint Date
MS Task Unique ID	—	MS Task Unique ID	Created Time
Object ID	—	Object ID	Deleted Time
Parent Task ID	—	Parent Task ID	Estimated Finish
Percent Complete	—	Percent Complete	Estimated Start
Task Duration	—	Task Duration	Last MS Project Update
Task Number	—	Task Number	Last Work/Dependency Update
Version	—	Version	Modified time

Topic Chart Fields

This table lists the available fields for topic charts.

Simple	Distribution	Correlation	Time-series
Attachment Count	Create charts based on fields currently being grouped.	Attachment Count	Created Time
Children Count	—	Children Count	Configuration Time
Recipient Count	—	Recipient Count	Deleted Time
Object ID	—	Object ID	Modified Time
Parent Topic ID	—	Parent Topic ID	—
Topic Number	—	Topic Number	—
Version	—	Version	—

Audit Chart Fields

This table lists the available fields for audit charts.

Simple	Distribution	Correlation	Time-series
Not Available	Create charts based on fields currently being grouped.	Not Available	Created Time
—	—	—	Deleted Time
—	—	—	Modified Time

Configuring Chart Colors

You can control the color of series data in the StarTeam **Chart** window.

1. Click **Edit Colors** on the **Chart** window toolbar.

   The **Edit Colors** dialog box opens.
2. Check **Use Custom Colors**.
3. Select a series from the list and click **Edit**,

   Note: You can edit only one series at a time.
4. Choose the color for the series in the **Select Color** dialog box using the swatches of color or the HSB or RGB values.
5. Click **OK**.
6. Select another series to edit its color.
7. Click **OK**.

Customizing Chart Titles

When you create a chart, you can insert titles for the top, left, right, or bottom of the chart. These titles can serve as overall titles, or names for the axes.

1. Click **Edit Titles** on the **Chart** window toolbar.
2. Type or edit the contents of the **Top**, **Left**, **Right**, or **Bottom** text boxes.
3. Click **OK**.

Exporting a Chart as an Image

You can export a chart as a `.jpg` image from the **Chart** window

1. Click the **Save Chart As** button on the **Chart** window toolbar. The **Save As** dialog box opens.
2. Type a name for the file.
3. Browse to or type a path to the target location for the `.jpg` file.
4. Click **Save**.

Generating Correlation Charts

This procedure explains how to generate a **Correlation** chart from the item data in the upper pane. A **Correlation** chart displays as a scatter chart, based on the fields you specify for the x-axis and y-axis.

You can filter out data in the upper pane to display only the data that you want to include in your chart. In addition, you can select specific items from the filtered data to include in your chart. A maximum of 60 fields can be displayed in the upper pane.

Note: You can only generate a **Distribution** or **Time-series** chart for audit entries

1. Click a component tab in the upper pane and select an item.
2. Optionally, select the specific items you want to chart.
3. Right-click in the upper pane and choose **Charts** > **Correlation** . The **Correlation Chart** dialog box opens.
4. Type a name for the chart in the **Chart Name** field.
5. Select one axis label from the **x-coordinates** list box, and one from the **y-coordinates** list box.

   Note: These coordinate lists display the names of the fields that are displayed in the upper pane that can be used as axes. A maximum of sixty fields can be displayed in the upper pane.
6. Select a printer page orientation: **Portrait** or **Landscape**.
7. Click **OK**.

Note: If the chart has too much data on it to be readable, increase the size of the chart window, or decrease the number of items in the chart.

Generating Distribution Charts

This procedure explains how to generate a **Distribution** chart of items grouped in the upper pane. A **Distribution** chart displays in the form of a pie chart. Each wedge indicates what fraction of the whole a group represents.

You can filter out data in the upper pane to display only the data that you want to include in your chart. In addition, you can select specific items from the filtered data to include in your chart. A maximum of 60 fields can be displayed in the upper pane.

Note: You can only generate a **Distribution** or **Time-series** chart for audit entries

1. Click a component tab in the upper pane and select an item.
2. Sort/Group the data on the selected tab in the upper pane.
3. Right-click in the upper pane and choose **Charts** > **Distribution** . The **Distribution Chart** dialog box opens.
4. Type a name for the chart in the **Chart Name** field.

5. Select which data to use for generating the chart. If you have selected specific items in the upper pane to chart, select **Current Selection**. Otherwise, use **Select All**.
6. Select a printer page orientation: **Portrait** or **Landscape**.
7. Click **OK**.

Note: If the chart has too much data on it to be readable, increase the size of the chart window, or decrease the number of items in the chart.

Generating Simple Charts

This procedure explains how to generate a **Simple** chart from the item data displayed in the upper pane. A **Simple** chart displays in a column format by default. However, the chart utility allows you to display simple charts in a variety of formats.

You can filter out data in the upper pane to display only the data that you want to include in your chart. In addition, you can select specific items from the filtered data to include in your chart. A maximum of 60 fields can be displayed in the upper pane.

Note: You can only generate a **Distribution** or **Time-series** chart for audit entries

1. Click a component tab in the upper pane and select an item.
2. Optionally, select the specific items you want to chart.
3. Right-click in the upper pane and choose **Charts** > **Simple** . The **Simple Chart** dialog box opens.
4. Type a name for the chart in the **Chart Name** field.
5. Select one or more fields from the **Series** list.

   Use **Ctrl+Click** or **Shift+Arrow** to select multiple fields. The dialog box lists the fields in the upper pane that can be charted. A maximum of sixty fields can be displayed in the upper pane.

   Note: The **Series** list can only contain fields that are number based. To add fields to the **Series** list, you must include a number-based field in the view/tab you want to chart.
6. Select which data to use for generating the chart. If you have selected specific items in the upper pane to chart, select **Current Selection**. Otherwise, use **Select All**.
7. Select a printer page orientation: **Portrait** or **Landscape**.
8. Click **OK**.

Note: If the chart has too much data on it to be readable, increase the size of the chart window, or decrease the number of items in the chart.

Generating Time-series Charts

This procedure explains how to generate a **Time-series** chart from the item data displayed in the upper pane. A Time-series chart displays as a line chart showing the number of items that have the same day, week, or month in the specified time/date field.

You can filter out data in the upper pane to display only the data that you want to include in your chart. In addition, you can select specific items from the filtered data to include in your chart. A maximum of 60 fields can be displayed in the upper pane.

1. Click a component tab in the upper pane and select an item.

   You can generate time-series charts for all item types.

   Note: A time-series chart must be based on a time/date field. If the upper pane displays no time/date fields, the application displays an error message. To avoid this error, add a time/date field to the columns in the upper pane.
2. Optionally, select the specific items you want to chart.

3. Right-click in the upper pane and choose **Charts** > **Time-series** to open the **Time-series Chart** dialog box.
4. Type a name for the chart in the **Chart Name** field.
5. Select a **Tracking Interval**: **Daily**, **Weekly**, or **Monthly**
6. Select one or more items from the **Time Series** list.
7. Check any additional options you want. You can specify a date range (**Limit To Period Of Time**), display accumulated items (**Cumulative Totals**), or include all date ranges in the chart (**Include Non-Represented Dates**).
8. Select which data to use for generating the chart. If you have selected specific items in the upper pane to chart, select **Current Selection**. Otherwise, use **Select All**.
9. Select a printer page orientation: **Portrait** or **Landscape**.
10. Click **OK**.

Note: If the chart has too much data on it to be readable, increase the size of the chart window, or decrease the number of items in the chart.

Working with Charts

The StarTeam **Chart** window displays the majority of charts by default in 3D, but you also have the option to toggle the 3D/2D view. You can zoom in and out, and rotate a chart on its x-y-z axes for all charts that you can view in 3D. The majority of charts shown in 2D do not allow for zoom or rotate operations.

Toggling a 3D and a 2D Chart

1. Generate a chart and display it in the **Chart** window.
2. Click the **Toggle 3D View** button on the **Chart** window toolbar.
3. To return to a 3D chart, click the **Toggle 3D View** button again.

Zooming in on a Chart

1. Generate a chart and display it in the **Chart** window.
2. Be sure that the chart type selected is in a view that allows for zooming.
3. While holding down the **Ctrl** key, click and hold down the left mouse button and move your mouse from left to right to zoom in and out on the chart.

Displaying the Chart Data on which the Table is Based

1. Generate a chart and display it in the **Chart** window.
2. Click **Toggle Table** on the **Chart** window toolbar. The chart data displays in a matrix that shows the legend entries vertically and the charted values for each of the file entries horizontally.
3. Click **Toggle Table** again to return to the chart display.

Displaying a Legend on a Chart

1. Generate a chart and display it in the **Chart** window.
2. Click **Toggle Legend** on the **Chart** window toolbar. Since the default is to display the legend, this hides the legend.
3. To display the legend, click **Toggle Legend** again.

Displaying a Horizontal or Vertical Grid on a Chart

1. Generate a chart and display it in the **Chart** window.

2. Click the **Horizontal Grid** button or the **Vertical Grid** button on the toolbar. Since the horizontal and vertical grid lines are displayed by default, this action hides the grids.
3. Click **Horizontal Grid** or **Vertical Grid** again to add the grid lines back to the chart display.

Default Chart Views and Zoom/Rotate Capabilities

Not all charts are capable of a 3D display, nor can you zoom in and out of, or rotate, all types of charts. For example, you cannot zoom or rotate any chart while it is in a 2D view.

The table below displays all the chart types and lists their default and available views, and their zooming and rotating capabilities.

Chart	3D - Zoom, Rotate	2D - Zoom, Rotate	Default View
Area	Yes	No	3D
Bar	Yes	No	3D
Column	Yes	No	3D
Bubble	No	No	2D (3D not available)
Heat Map	Yes	No	3D
Line	Yes	No	3D
Pie	Yes	No	3D
Scatter	No	No	2D (3D not available)
Stack Bar	Yes	No	3D
Stack Column	Yes	No	3D

Index