Timely Migration

User Guide

Chapter 1: Getting Started

Installation

Requirements

The Timely Migration software requires the following prerequisites to be installed on the machine running the migration:

- .NET 3.5 framework
- .NET 4.0 framework
- Visual Studio Team System 2010 Team Explorer

Compatibility for TFS 2008 is handled through the Visual Studio Team System 2010 Team Explorer client.

Additionally the Timely Migration software requires access to a SQL or SQL Express installation to create a staging database. The database engine may be hosted on a machine separate from the machine running the Timely Migration software. In order to be able to connect to the database, it must be configured to accept local and remote connections using both TCP/IP and named pipes.

Different Timely Migration modules will also have additional prerequisites:

CVSToTFS

- CVSNT/WinCvs

StarTeamToTFS

- StarTeam 2008 Release 2 SDK Runtime for .NET 10.4

Installing the software

1. Validate that the prerequisites are installed.
2. Install the Timely Migration MSI

Migration Planning

History is important because it contains a record of what changes were made when, and that can be used to try to figure out why changes were made and what defects they were originally intended to fix. This allows future developers to consider why the code works the way it does to help keep them from reintroducing old defects and causing costly recycles.

However when migrating history, accidental changes and quirks with the way the original source control system was used will often show up during the migration phase as discrepancies that will need to be reconciled. If the original source control database wasn't regularly maintained, database corruption could also require reconciliation.

Because of the need for reconciliation, it should be expected that you will need to plan for time to watch the migration to make sure it finishes and to examine what was migrated. You will need to determine whether problems resulted from the migration process or if the problem lies with the origin source database, and if those problems are important and need to be resolved or can be skipped.

For example certain version control systems may treat files with the same name but difference cases as separate files. With TFS these files are treated as the same file. When migrating changes you may have files ‘a’ and ‘A’ get renamed to ‘B’ and ‘C’. Since in TFS there is only one file A it cannot be renamed twice and a migration error will result and need to be reconciled.

When determining if anything needs to be reconciled, it is recommended that you use a non production server as your migration target. This prevents the production server from being filled with files that will never be used but still take up database space and be subject to periodic maintenance. Once reconciliation has been performed and you are happy with the results you can rerun the migration with the production server as the migration target.

Chapter 2: Migration Concepts

Sessions

A session consists of the configuration which defines what will be migrated as well as the state of what is targeted for migration. The configuration portion defines what code will be migrated and where in the target system it will be migrated to. A session also contains settings which define what labels/tags will be migrated.

Session state is stored in a SQL staging database and contains the list of items the configuration indicated should be migrated as well as whether or not those items have been migrated yet. This allows a migration to be interrupted and then pick up where it left off. This also allows a migration to be performed in phases and allow development to continue in the origin source database while the target source database becomes ready for production.

Mappings

Mappings define the location of the code in the origin source database that you want to migrate and the destination where you want the code to be located in the target source database. For a simple migration, there may only be one mapping which defines the source and destination for a single branch, or the root of the origin source database if you want to migrate an entire database.

Mappings can also be made more complex if you want to use the migration process to help you reorganize the source tree. For example your source tree may have started out with release branches in the same location as the main development branch, and over time the source tree may have gotten cluttered with many release branches. You can use the migration effort as a chance to clean up the tree and use the mapping functionality to help you move where each individual branch get migrated to. Obsolete source can also be removed from a migration by creating a mapping that cloaks an individual folder.

Depending on the origin source control system used, reorganizing the source tree may be a necessity. TFS has limitations on the number of characters allowed in a path, 248 characters for a directory and 260 characters for a file, so if the path depth from the origin system combined with where you want it to live in TFS causes the limits to be exceeded, mappings can be used to rename or relocate folders to be able to meet those limits. More information on TFS path limitations can be found on MSDN here.

Sometimes attempts to clean up the source tree in the origin source database by moving or renaming folders result in artifacts being left behind in the old directory names. Mappings can be used to help correct these issues by configuring all operations in the old directory to be migrated to the renamed directory.

Certain source control systems like CVS store items for different branches in the same location in the source tree, i.e. “/Foo/Bar/Baz.c”, and use another identifier such as a branch tag to help specify what branch the file is supposed to correspond to. For source control systems like CVS, the mappings will be used to help define what branches will be migrated and what their destination locations will be.

Phases

When converting from one source control system to another there are two phases to a session, analysis and migration. During the analysis phase the session configuration is applied to the origin source database and items to be migrated are identified and recorded in a SQL staging database along with their source and destination locations. After the analysis phase completes, the migration phase starts and reads the records in the SQL staging database to determine what it needs to move to the target source control system.

If a session is restarted, it starts by running the analysis phase to see if any new data has been added to the origin source control system and then proceeds to the migration phase once the analysis phase is over.

Chapter 3: Configuration

Selecting What to Migrate

The first part of configuration is deciding what to migrate. This is an important step because you not only need to consider what you want to migrate right now, but what features of Team Foundation Server you might want to use later and what you would need to migrate in order to use them.

For instance, if you want to migrate a single release branch, but know that sometime later you will want to use the built in branching/merging functionality of Team Foundation Server, you should plan to migrate the main development branch the release branch was branched off of.

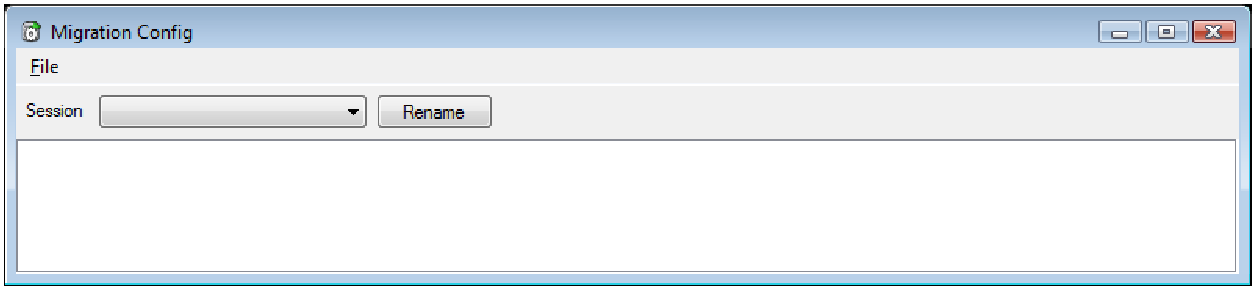
If you have a source database that hosts multiple products, you should think about what Team Projects you want them to be migrated to and how you manage labels. Labels within TFS are global to a team project. If you have multiple products which each started off on a version 1.0 release and they have labels corresponding to build numbers that are unique to the product, but conflict across products, they will conflict in the Team Foundation Server "Find Label" UI and you'll find your different products attributed to the same label.

To help prevent label conflicts there are two main options. The first is that you could move each product to a separate Team Project. Due to the performance considerations of having many Team Projects as well as user maintenance of each Team Project you may still want to locate the different products in the same Team Project. If that is the case you can migrate each product with a separate session and use the label prefixing functionality to add a prefix specific to the product.

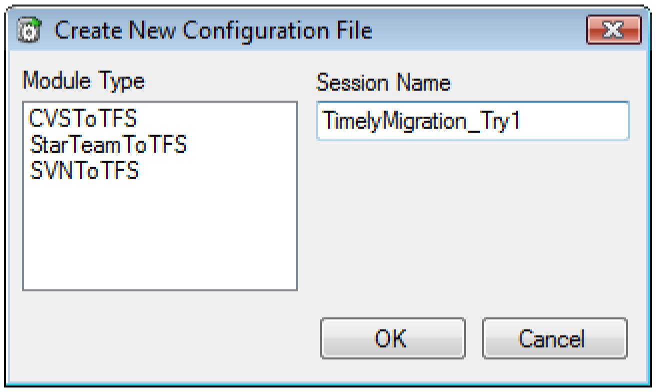
Selecting what to migrate also includes what to ignore. Often times the source tree may be used to store large binary files that are loosely related to the product, but are not directly used in the official build process, such as SDK installs, sample databases or even documentation. Since these files will take up hard drive space on client machines and increase the overall time required to perform a get from TFS, it may be a good idea to prune them from the tree using cloaked folder mappings.

Configuring a New Session

Once you have an idea of what you want to migrate it is time to use the MigrationConfig.exe application to create a new configuration file for a session with the settings you want. To start, launch MigrationConfig.exe and you should see the following form:



Select File->New to create a new configuration file. At this point you will be prompted to select some basic information for the session as shown in the following dialog:



Under "Module Type" select the Timely Migration module that corresponds to your origin source control system. "Session Name" should be filled in with something that corresponds to what you want to migrate. If you are unsure of exactly what you want to migrate and think you may be performing multiple migrations, it is a good idea to add an identifier that indicates which iteration you are on.

Once these fields are filled out, select OK to start a new configuration file. At this point you should be brought back to the main form and new tabs should exist for the various types of information that you can configure for the selected Timely Migration module. Each of these tabs is described later in the Session Configuration section.

If the prerequisites for the module that are you are trying to configure are not installed on the system you will get an error message identifying the prerequisite that appears to be missing. You will still be able to configure a session without having the prerequisites installed on the system, but attempts to perform the migration will fail.

To save the configuration file, select either File->Save or File->Save As or press Ctrl-S.

Updating Session Configuration

During the migration process there may be configuration settings that you will want to adjust such as reorganizing where branches are migrated to or what content should be dropped out of the migration.

Note: When changing certain configuration settings such as folder mappings, errors can result if the same session is restarted and the paths after the restart no longer match the paths used before the restart. To prevent issues make sure to use a new session name.

To edit an existing configuration, launch MigrationConfig.exe, select File->Load to bring up an open file dialog and then select the configuration file you want to edit.

If the prerequisites for the module that are you are trying to edit are not installed on the system you will get an error message identifying the prerequisite that appears to be missing. You will still be able to edit a session without having the prerequisites installed on the system, but attempts to perform the migration will fail.

If you have already performed a migration, it should be noted that the staging database which stores session state will contain information about the current session. This session state is normally used to continue a migration where it was left off, but if you want to perform a new migration from the beginning, it will have the effect of preventing the migration.

In order to be able to run a new migration, you should either rename the session by pressing the "Rename" button next to the "Session" box or drop the staging database. Renaming the session used in conjunction with a setting to prefix labels with the session name has the benefit that the labels will not get overcrowded with files from multiple sessions. If the staging database is getting large and SQL Express is being used as the database engine, you may want to drop the staging database to prevent database size restrictions from being enforced.

Note: If you have performed a migration to a production server, the staging database will contain information that you may need if you ever want to continue a migration to pick up changes after an initial migration or if something in the initial migration needs to be repaired. To allow for these operations in the future the database should be backed up and archived before it is dropped.

In addition to state stored in the staging database, if the session had previously completed, TFS will contain files and labels that correspond to the current session. To prevent conflicts with files, you can either change the destination for files to as described in the Folder Mappings section so that they are different for the new migration or move/delete the folders within TFS.

Moving files within TFS over deleting files in TFS has the added benefit that you can more easily look at file history if you are trying to compare results from the different migration attempts.

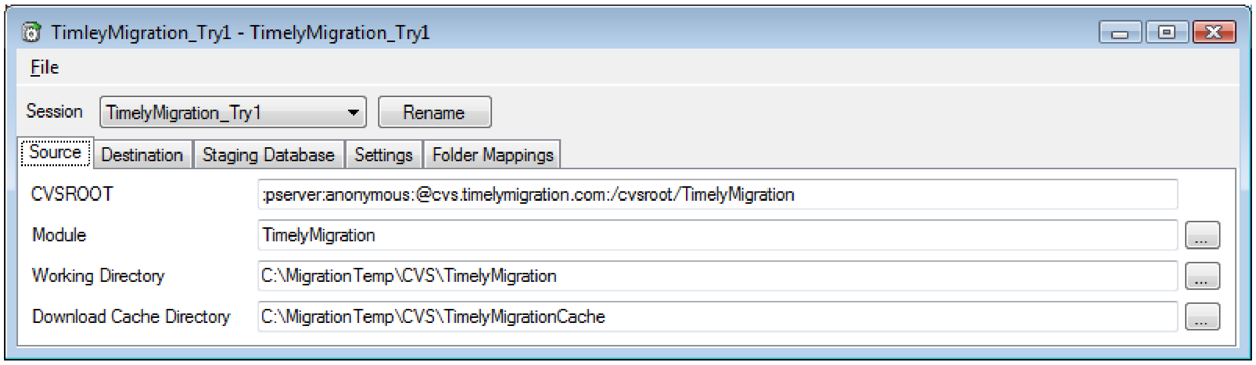
To prevent conflicts with labels, labels can be automatically prefixed with either the session id or a specific prefix and is described more in the Settings section.

Session Configuration

Source

The “Source” tab contains information related to where the source that you want to migrate is located such as connection information for the source control database. As the connection information will be different across various source control systems, the contents of the “Source” tab will be tailored to each source control system.

CVSToTFS

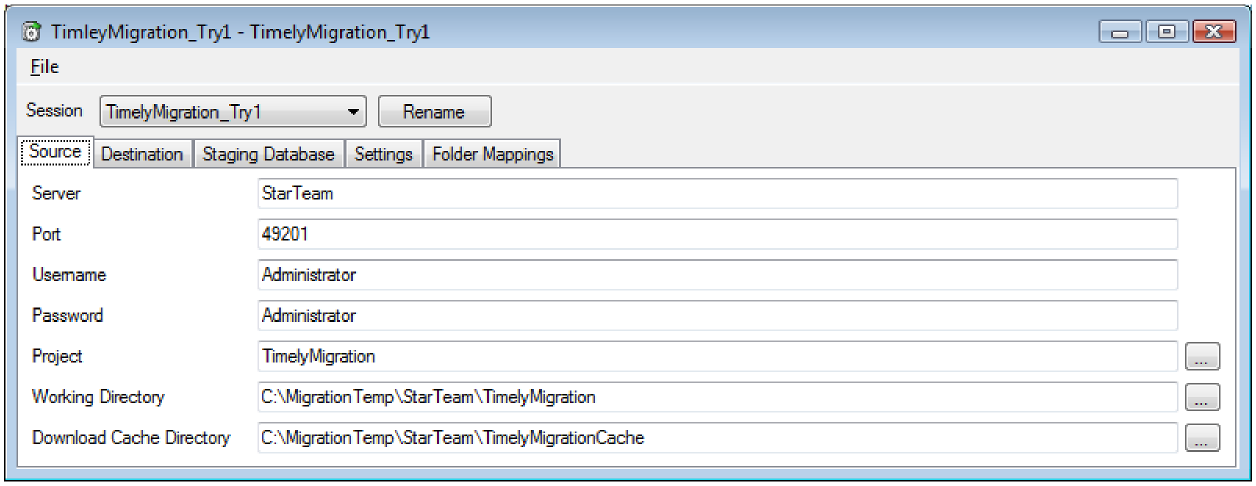


CVSROOT :pserver:anonymous:@cvs.timelymigration.com:/cvsroot/TimelyMigration
Module TimelyMigration
Working Directory C:\MigrationTemp\CVS\TimelyMigration
Download Cache Directory C:\MigrationTemp\CVS\TimelyMigrationCache

Descriptions for each of the fields are as follows:

Setting Name	Meaning
CVSROOT	This setting stores the CVSROOT string that CVS and WinCVS use to locate the CVS database and help determine what type of authentication, user name, and password to use.
Module	This setting stores the name of the CVS module that you want to migrate source from.
Working Directory	This setting stores the name of a directory to be used with retrieving files from the origin source control system. The migration will overwrite all files in this directory so it should be a directory dedicated to the migration.
Download Cache Directory	This setting stores the name of a directory to be used to cache individual files revisions. If repeated migrations are run, this setting can speed up the migration greatly if the database is a remote database. Additionally if there are problems with individual file revisions on the server, the file in the cache can be updated to hold a correct revision.

StarTeamToTFS



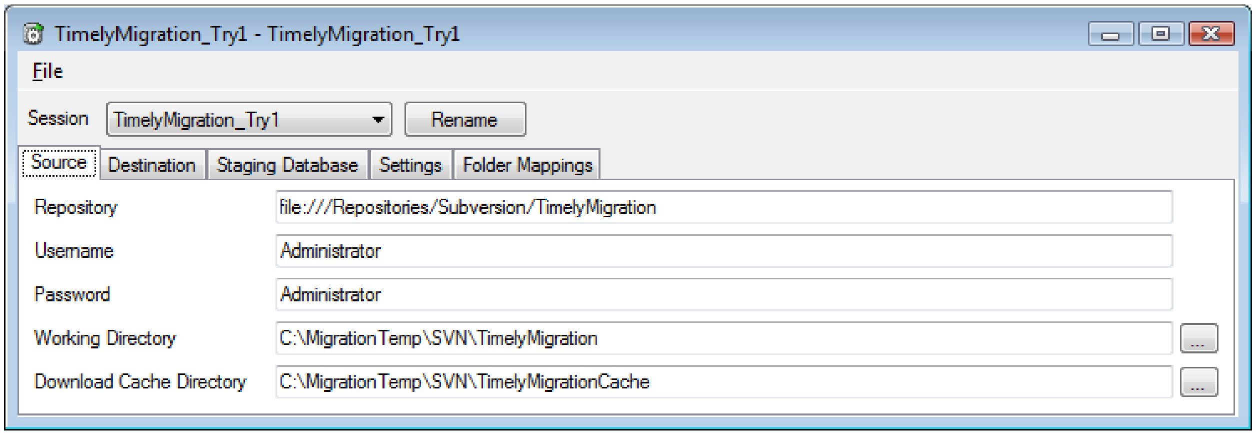
Session: TimelyMigration_Try1

Server	StarTeam
Port	49201
Username	Administrator
Password	Administrator
Project	TimelyMigration
Working Directory	C:\MigrationTemp\StarTeam\TimelyMigration
Download Cache Directory	C:\MigrationTemp\StarTeam\TimelyMigrationCache

Descriptions for each of the fields are as follows:

Setting Name	Meaning
Server	This setting stores the name of the StarTeam server to connect to.
Port	This setting stores the port that the StarTeam server is listening on.
Username	Name of the user that should be used to connect to the StarTeam server.
Password	Password for the user that should be used to connect to the StarTeam server.
Project	This setting stores the name of the StarTeam project that you want to migrate source from.
Working Directory	This setting stores the name of a directory to be used with retrieving files from the origin source control system. The migration will overwrite all files in this directory so it should be a directory dedicated to the migration.
Download Cache Directory	This setting stores the name of a directory to be used to cache individual files revisions. If repeated migrations are run, this setting can speed up the migration greatly if the database is a remote database. Additionally if there are problems with individual file revisions on the server, the file in the cache can be updated to hold a correct revision.

SVNToTFS



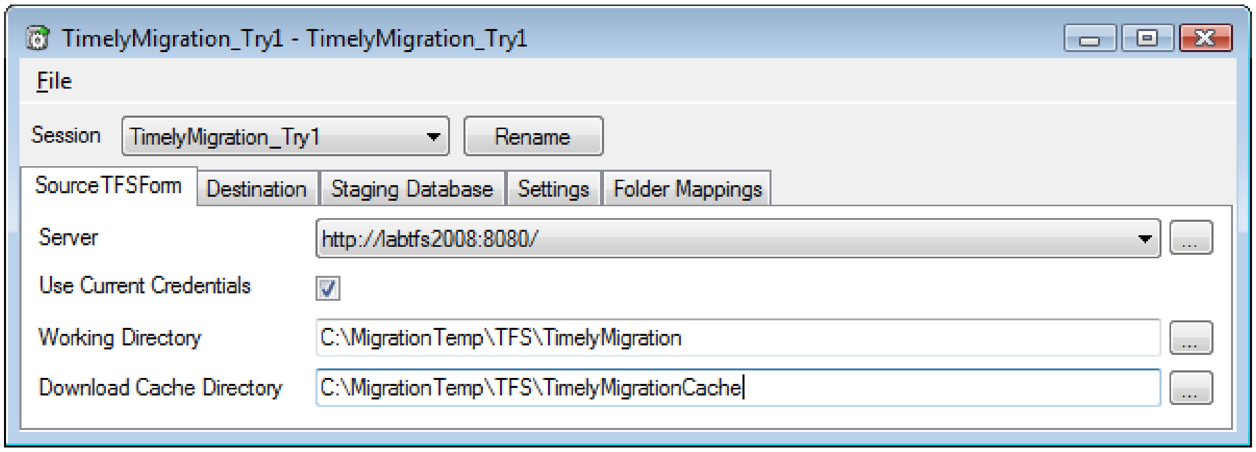
Session: TimelyMigration_Try1

Repository	file:///Repositories/Subversion/TimelyMigration
Username	Administrator
Password	Administrator
Working Directory	C:\MigrationTemp\SVN\TimelyMigration
Download Cache Directory	C:\MigrationTemp\SVN\TimelyMigrationCache

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
Repository	This setting stores the location of the Subversion repository to connect to.
Username	Name of the user that should be used to connect to the Subversion server.
Password	Password for the user that should be used to connect to the Subversion server.
Working Directory	This setting stores the name of a directory to be used with retrieving files from the origin source control system. The migration will overwrite all files in this directory so it should be a directory dedicated to the migration.
Download Cache Directory	This setting stores the name of a directory to be used to cache individual files revisions. If repeated migrations are run, this setting can speed up the migration greatly if the database is a remote database. Additionally if there are problems with individual file revisions on the server, the file in the cache can be updated to hold a correct revision.

TFSToTFS



Session: TimelyMigration_Try1

Server: http://labtfs2008:8080/

Use Current Credentials: ✓

Working Directory: C:\MigrationTemp\TFS\TimelyMigration

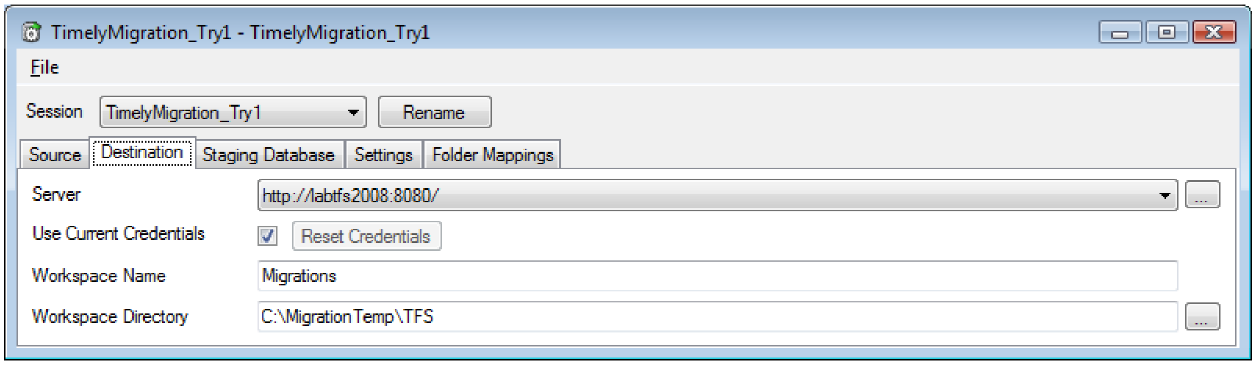
Download Cache Directory: C:\MigrationTemp\TFS\TimelyMigrationCache

Descriptions for each of the fields are as follows:

Setting Name	Meaning
Server	URL of the TFS server to migrate changes to.
Use Current Credentials	This flag indicates whether to use the credentials for the current user when accessing TFS or to prompt for and cache a separate set of credentials. If checked the credentials of the user that runs the migration will be used to access TFS. If cached credentials are used the “Reset Credentials” button will become active and will let you clear out any credentials that have been stored. This can be used to reset a password if it was changed in the TFS environment after it was cached.
Working Directory	This setting stores the name of a directory to be used with retrieving files from the origin source control system. The migration will overwrite all files in this directory so it should be a directory dedicated to the migration.
Download Cache Directory	This setting stores the name of a directory to be used to cache individual files revisions. If repeated migrations are run, this setting can speed up the migration greatly if the database is a remote database. Additionally if there are problems with individual file revisions on the server, the file in the cache can be updated to hold a correct revision.

Destination

The “Destination” tab contains information about where the source should be migrated to. Since the destination will always be a TFS server, the “Destination” tab will be the same across all Timely Migration modules and will look like the following:



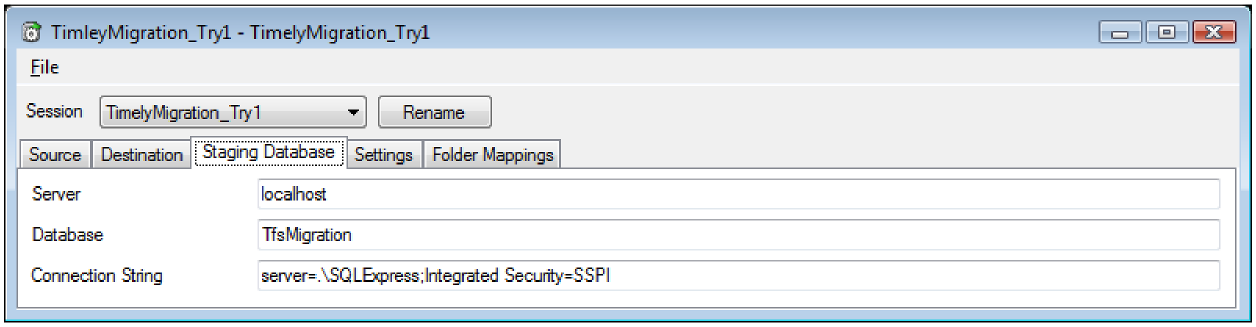
Session: TimelyMigration_Try1
Server: http://labtfs2008:8080/
Use Current Credentials: ✓
Workspace Name: Migrations
Workspace Directory: C:\MigrationTemp\TFS

Descriptions for each of the fields are as follows:

Setting Name	Meaning
Server	URL of the TFS server to migrate changes to.
Use Current Credentials	This flag indicates whether to use the credentials for the current user when accessing TFS or to prompt for and cache a separate set of credentials. If checked the credentials of the user that runs the migration will be used to access TFS. If cached credentials are used the “Reset Credentials” button will become active and will let you clear out any credentials that have been stored. This can be used to reset a password if it was changed in the TFS environment after it was cached.
Workspace Name	This setting is used to identify the TFS workspace to be used with the migration. Files in this workspace will be repeatedly cleared out and updated so it is recommended that a dedicated workspace for the migration is used.
Workspace Directory	This setting is used to identify the folder that the workspace will be mapped to on disk. Files in this folder will be repeatedly cleared out and updated so it is recommended that a dedicated folder for the migration is used.

Staging Database

The “Staging Database” tab contains connection information for the SQL database used to store session state. As each Timely Migration module makes use of the staging database, the “Staging Database” tab will be the same across all Timely Migration modules and will look like the following:



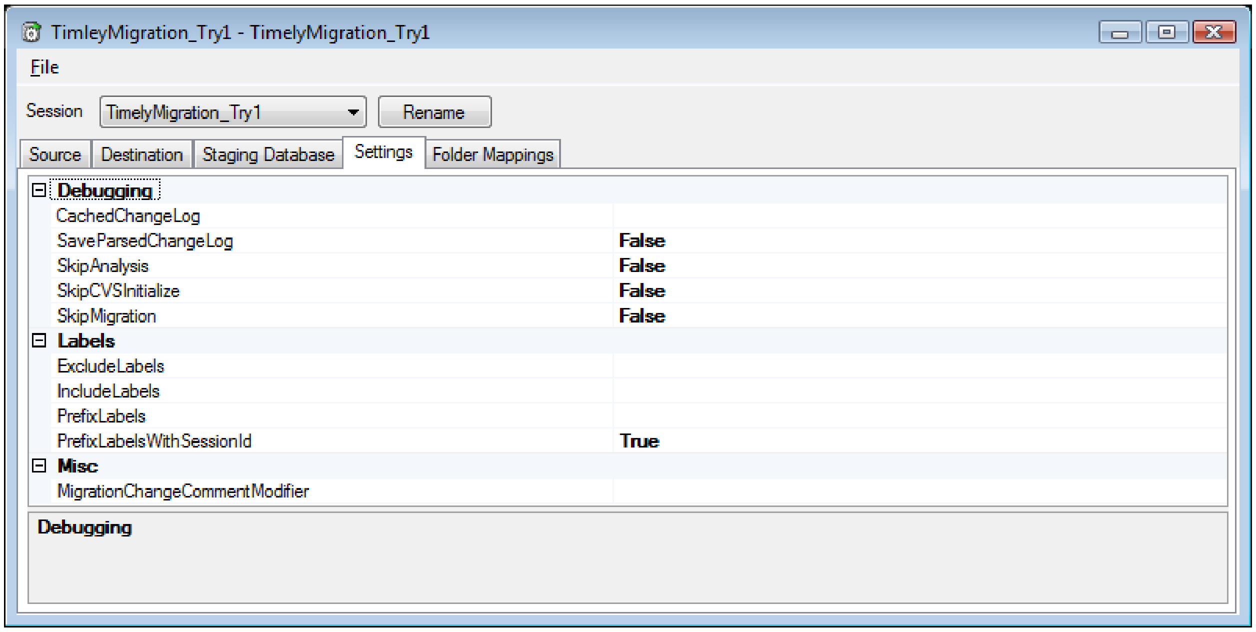
Server: localhost
Database: TfsMigration
Connection String: server=.\SQLExpress;Integrated Security=SSPI

Descriptions for each of the fields are as follows:

Setting Name	Meaning
Server	This setting is used to identify the server address to connect to.
Database	This setting is used to identify the individual database to be used.
ConnectionString	This setting is used to identify the SQL connection string to be used when connecting to the staging database. Take care to make sure that the correct SQL Server instance is used.

Settings

The “Settings” tab stores various options for how source control data should be migrated. As different source control systems have different characteristics, these options will change based on the source control system. The “Settings” tab will look like the following although not all modules will show all fields:



Debugging	
CachedChangeLog	
SaveParsedChangeLog	**False**
SkipAnalysis	**False**
SkipCVSInitialize	**False**
SkipMigration	**False**
Labels	
ExcludeLabels	
IncludeLabels	
PrefixLabels	
PrefixLabelsWithSessionId	**True**
Misc	
MigrationChangeCommentModifier	

Debugging

Common

Descriptions for the settings common between Timely Migration modules are as follows:

Setting Name	**Meaning**
SkipAnalysis	This setting can be used to disable analysis. Configure this setting to “true” to skip the analysis phase. This setting can be used when troubleshooting issues in the migration phase to skip over the time it takes to perform an analysis.
SkipMigration	This setting can be used to disable migration. Configure this setting to “true” to skip the migration phase. This setting can be used when troubleshooting issues in the analysis phase to prevent any migration from occurring with analysis data that you want to examine.
ItemExistsHandler	This setting is used to determine what action to take if an Add operation is being attempted on an item that already exists. “Fail” – This error will cause the migration to fail. This is the default option if a value is not specified. “Ignore” – A warning message will be logged and the migration will continue. “ChangeToEdit” – The operation will be changed to an Edit operation.
PrefixLabels	This setting is used to add a prefix to any labels that are migrated. Configure this setting to the prefix that you wish to use. If multiple projects have similar version schemes and are migrated to the same TFS project, this setting can be used to prevent label name conflicts.
PrefixLabelsWithSessionId	This setting is used to add a prefix to any labels that are migrated. Configure this setting to “true” to automatically prefix labels. If multiple source projects have similar version schemes and are migrated to the same TFS project, this setting can be used to prevent label name conflicts. Additionally if you are performing multiple test migration attempts, this setting can be used to automatically change the label name based on the session id to prevent conflicts.
IncludeLabels	This setting is used to filter labels that will be migrated. Only labels matching the label filter will be migrated. The format is a list of semi-colon delimited System.Text.RegularExpressions.Regex expression strings. Filtering for labels in either a 1.1 or 1.2 version would be represented by “1\.1;1\.2”.

Setting Name	Meaning
ExcludeLabels	This setting is used to filter labels that will be migrated. Labels matching the label filter will not be migrated. The format is a list of semi-colon delimited System.Text.RegularExpressions.Regex expression strings. Filtering out labels in either a 1.1 or 1.2 version would be represented by “1\.1;1\.2”.
MigrationChangeCommentModifier	This setting is used to override the text that is appended to the end of each checkin comment to indicate that the change was made as part of an automated migration process. Keywords can be used to reformat the entire comment string. The currently supported keyword are: $(DateTime) - Original change timestamp $(Comment) - Original change comment $(Name) - The name used to track the change. Will translate to a source id if supported.
LoggingMode	This setting can be used to change the level of detail logging provides. 'Normal' - Regular log messages will be displayed. 'Verbose' - Detailed log information will be displayed. Note that setting logging to verbose may increase the size of the log file dramatically.
UserMappingFile	This setting can be used to set the name of the user mapping file to be used with the migration. If a file is not specified the user mapping file will default to "<SessionId>UserMappings.xml".
SessionContinuationWarning	This setting is used to alert the user if they are continuing a session that was previously run. Configuration changes will not be reflected in previously analyzed or migrated data, so if a user is unaware that they are continuing a session the warning alerts the user that they may not see the changes they expect.
DropLongPaths	This setting can be used to drop out items that are causing errors by having a server path or a local path that exceeds the maximums defined by Windows. Typically local paths can be made shorter by changing the Workspace Directory used by TFS in the Destination tab.

CVSToTFS

Descriptions for the settings specific to CVSToTFS are as follows:

Setting Name	Meaning
CachedChangeLog	Set the value to the location of a CVS changelog that you want to use with the migration instead of querying a changelog from the server. This setting can be used when you want to customize what is migrated via the changelog. **Note:** If you use a cached changelog you will not get updates after the changelog was created. If you wish to use this setting it is a good idea to lock the CVS database so that no more changes can be made.
SaveParsedChangeLog	This setting is used to save out intermediate parsed forms of the migration data. Configure this setting to “true” to save out the data to the CVS working directory. This setting can be used when troubleshooting issues in the analysis phase.
SkipCVSInitialize	This setting is used to skip the initialization of the CVS working directory. Configure this setting to “true” to skip initialization including a CVS checkout. This setting can be used save time when performing repeated migrations.
CaseDifferenceHandler	This setting is used to determine what action to take if there are files in the same directory where the file names differ only by case. 'Ignore' - A warning message will be logged and the migration will continue. Errors may occur during the migration phase if the actions between the different files conflict. 'DropDuplicateFiles' - Files with duplicate names except for case will be dropped from the migration.
IgnoreDefaultVendorBranch	This setting is used to determine whether or not the default Vendor branch that is created when importing source into CVS is considered valid for operations like trying to determine which branch a tag should be applied to as the primary branch.

StarTeamToTFS

Descriptions for the settings specific to StarTeamToTFS are as follows:

Setting Name	Meaning
BlackListItems	This setting is used to indicate items that should not be processed at all. This can be used if the Borland StarTeam API repeatedly fails to access certain items to allow the problem items to be skipped and the rest of the migration to continue. The format of this setting is '<ItemId1>;<ItemId2>' where the item ids can be found in the log file directly after item names such as <Folder>\<Item>:<ItemId>.
DeletedItemDiscoveryMode	This setting is used to determine how StarTeamToTFS should look for deleted items if supported by StarTeam. Note only StarTeam releases 2006 and later support returning information about deleted items. 'RecycleBin' - Will use the StarTeam RecycleBin feature to ask the server for the deleted items. 'None' - Deleted items will not be migrated. This setting can be used if StarTeam has problems returning deleted item information. 'Scan' - Will scan for deleted items and will not use any cached information. 'PersistentScan' - Will scan for deleted items and will cache information about the files it has scanned between attempts. RecycleBin is the most efficient option. Scan is intended to be a troubleshooting option that scans through item ids to try to find deleted items. Because the operation is a scan it can be quite slow. PersistentScan behaves similar to Scan but will cache results between migration. If the same project is migrated twice, performance will be similar to that of RecycleBin. To improve initial performance, the DeletedItemExternalScanFile setting can be used to cache information manually extracted from the StarTeam database.
DeletedItemExternalScanFile	This setting is used together with DeletedItemDiscoveryMode to cache deleted item information manually extracted from the StarTeam database. If DeletedItemDiscoveryMode is set to PersistentScan, the data in the file identified by DeletedItemExternalScanFile will be cached. This data should

Setting Name	Meaning
	be in a comma separate file (.csv) with each line formatted like '<ViewID>, <ItemID>, <DeletedUserID>'.
ForceItemRevisionDisabled	This setting is ued to disable processing Item Revisions which store information such as if a item has been moved between folders. This setting can be use to fix files in place during a migration and prevent folder moves from being migrated.
LabelDiscoveryMode	This setting is used to determine how StarTeamToTFS should look for labels. 'LabelViewBased' - Will use label based view configurations to examine the files in a label. 'FileBased' - Will scan for labels that are attached to an item as the item is processed. LabelViewBased is the most efficient option, but sometimes StarTeam will report labels on files that it will not map in a view.
StandaloneBranchMode	This setting is used to how branches will be migrated if they are not being migrated with their parent branch. 'AllRevisions' - Will attempt to migrate all revisions that are visible in the view. 'BranchedRevisionsOnly' - Will only attempt to migrate revisions that were either part of the initial branch point or were added afterwards. AllRevisions can be used when attempting to use a child view to selectively migrate folders in the root view. BranchedRevisionsOnly can be used to migrate a repository in stages by migrating child views as needed into TFS on top of branches created in TFS.
StarTeamSDKCommunicationLogFile	This setting is used to configure the StarTeam SDK to enable it's logging around commands that are being sent to the StarTeam Server. To turn on logging, configure this setting to name of the file that you want the StarTeam SDK to use.

SVNToTFS

Descriptions for the settings specific to SVNToTFS are as follows:

Setting Name	Meaning
TagFolders	This setting is used to indicate which folders were used to tag releases or builds within Subversion and should be translated to labels within TFS. If multiple folders were used they should be separated with a ';'.
StartingChangeset	This setting is used to indicate the starting Subversion revision that should be examined as part of the migration. This can be used with advanced migrations to manually shape the content of what is migrated.
EndingChangeset	This setting is used to indicate the ending Subversion revision that should be examined as part of the migration. This can be used with advanced migrations to manually shape the content of what is migrated.
IgnoreKeywordSubstitution	This setting is used to indicate whether or not client side keyword substitutions should be maintained. If keyword substitutions are maintained, then keywords will be tracked and substituted across branching and renaming operations. When working with large remote repositories, querying for properties on a branch can be a time consuming operation taking 20 minutes or more to return. This setting can be used to disable keyword substitution to cut down on the amount of time needed to analyze a repository.
PrefetchChangesetCacheSize	This setting is used to indicate how many processed changesets will be stored in case a future changeset needs to reference it. Large cache sizes can be used to reduce the number of changesets that need to be reprocessed.
ConvertedChangesetCacheSize	This setting is used to indicate how many revisions should be retreived at once when scanning the source repository. Large cache sizes can be used with remote servers to reduce the number of individual calls that are made.

TFSToTFS

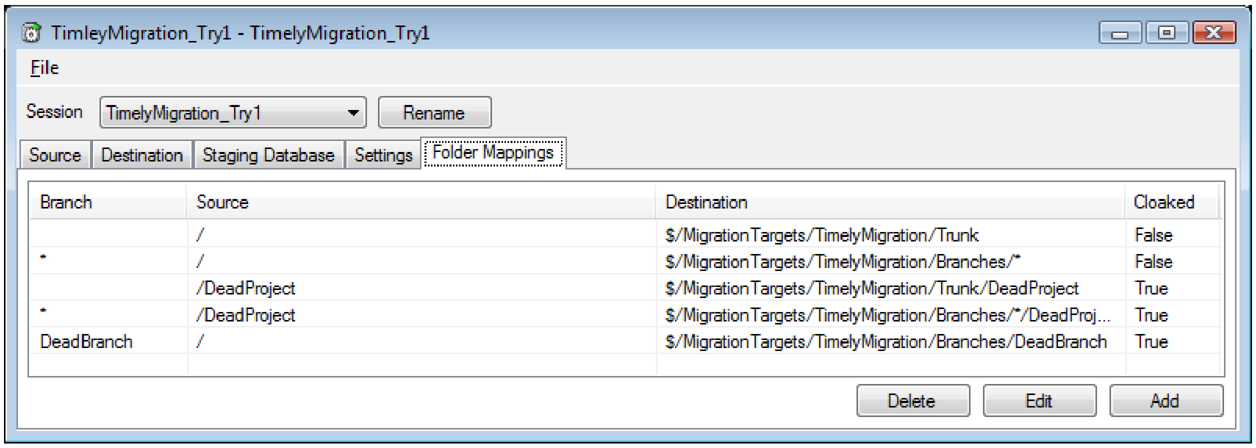
Descriptions for the settings specific to TFSToTFS are as follows:

Setting Name	**Meaning**
StartingChangeset	This setting is used to indicate the starting TFS changeset that should be examined as part of the migration. This can be used with advanced migrations to manually shape the content of what is migrated.
EndingChangeset	This setting is used to indicate the ending TFS changeset that should be examined as part of the migration. This can be used with advanced migrations to manually shape the content of what is migrated.

Folder Mappings

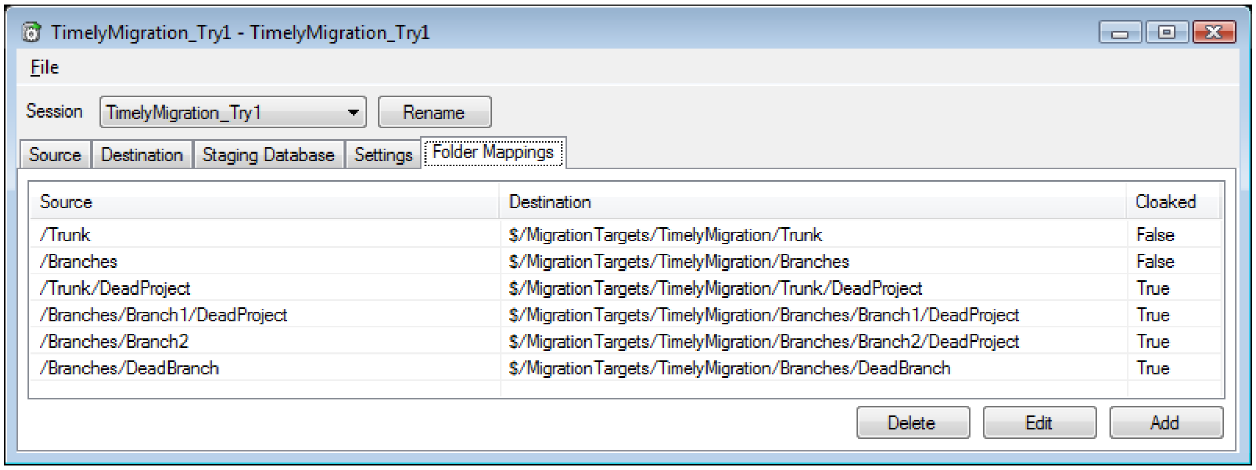
The “Folder Mappings” tab is used to customize what folders and branches will be part of the current session. Folder mappings can be similar across different source control systems, although differ based on whether the source control system has a special concept for a branch, or merges the concept of a branch into a folder path.

For systems that have a special concept for a branch, the “Folder Mappings” tab will look like the following:



Branch	Source	Destination	Cloaked
	/	$/MigrationTargets/TimelyMigration/Trunk	False
*	/	$/MigrationTargets/TimelyMigration/Branches/*	False
	/DeadProject	$/MigrationTargets/TimelyMigration/Trunk/DeadProject	True
*	/DeadProject	$/MigrationTargets/TimelyMigration/Branches/*/DeadProj...	True
DeadBranch	/	$/MigrationTargets/TimelyMigration/Branches/DeadBranch	True

For systems that do not have a special concept for a branch, the “Branch” column will be removed and the “Folder Mappings” tab will look like the following:



Source	Destination	Cloaked
/Trunk	$/MigrationTargets/TimelyMigration/Trunk	False
/Branches	$/MigrationTargets/TimelyMigration/Branches	False
/Trunk/DeadProject	$/MigrationTargets/TimelyMigration/Trunk/DeadProject	True
/Branches/Branch1/DeadProject	$/MigrationTargets/TimelyMigration/Branches/Branch1/DeadProject	True
/Branches/Branch2	$/MigrationTargets/TimelyMigration/Branches/Branch2/DeadProject	True
/Branches/DeadBranch	$/MigrationTargets/TimelyMigration/Branches/DeadBranch	True

Both examples demonstrate how to migrate a trunk and its branches along with pruning a project named “DeadProject” and branch named “DeadBranch” out of the migration.

For source control systems that support a special concept of a branch, the branches that you want to be migrated can be specified by using a wildcard indicator, “*” or the individual branch name. When a branch is specified by a wildcard indicator, the “Destination” field is expanded and the “*” is replaced with the branch name. For source control systems that do not support a special concept of a branch, multiple branches can be specified at once if they are under the same folder, such as “/Branches” in the second example as well as individually by specifying the full path of the branch folder.

In order to omit a folder from migration, the “Cloaked” field should be set to true and the “Source” field should be set to the path you want to omit. If the branch concept is supported, a folder across all branches can be easily specified by using the wildcard indicator. If the branch concept is not supported, each branch that you want to remove the folder from needs to be specified individually. While no data will be migrated for a “Cloaked” folder, the “Destination” field should still be set to the location it would be moved to prevent internal TFS consistency constraints when setting up the migration TFS workspace.

Omitting a branch from migration is similar between both types of systems as the branch should be individually specified to be removed.

Descriptions for each of the fields are as follows:

Setting Name	Meaning
Branch	Name of the branch that the entry corresponds to. “*” represents a wildcard indicator used to specify all branches.
Source	This setting is used to identify the folder in the origin source control database that you want to migrate. This folder should be unique. Path conventions in this setting should follow naming conventions for the origin source control system.

Setting Name	**Meaning**
Destination	This setting is used to identify the folder in TFS where you want the file in the src folder migrated to. This folder does not need to be unique if you want multiple folders to map to the same destination folder. Path conventions in this setting should follow TFS naming conventions. If a wildcard indicator is used as the Branch name, the Destination field should contain an "*" where the branch name should be included in the TFS path.
Cloaked	This setting is used to indicate whether or not you want to migrate this folder or leave it out. This can be used to prune dead branches that you do not want migrated. It can also be used to ignore individual folders that may contain content that you no longer want in your source tree such as SDK installs, sample databases or documentation.

Chapter 4: Performing a Migration

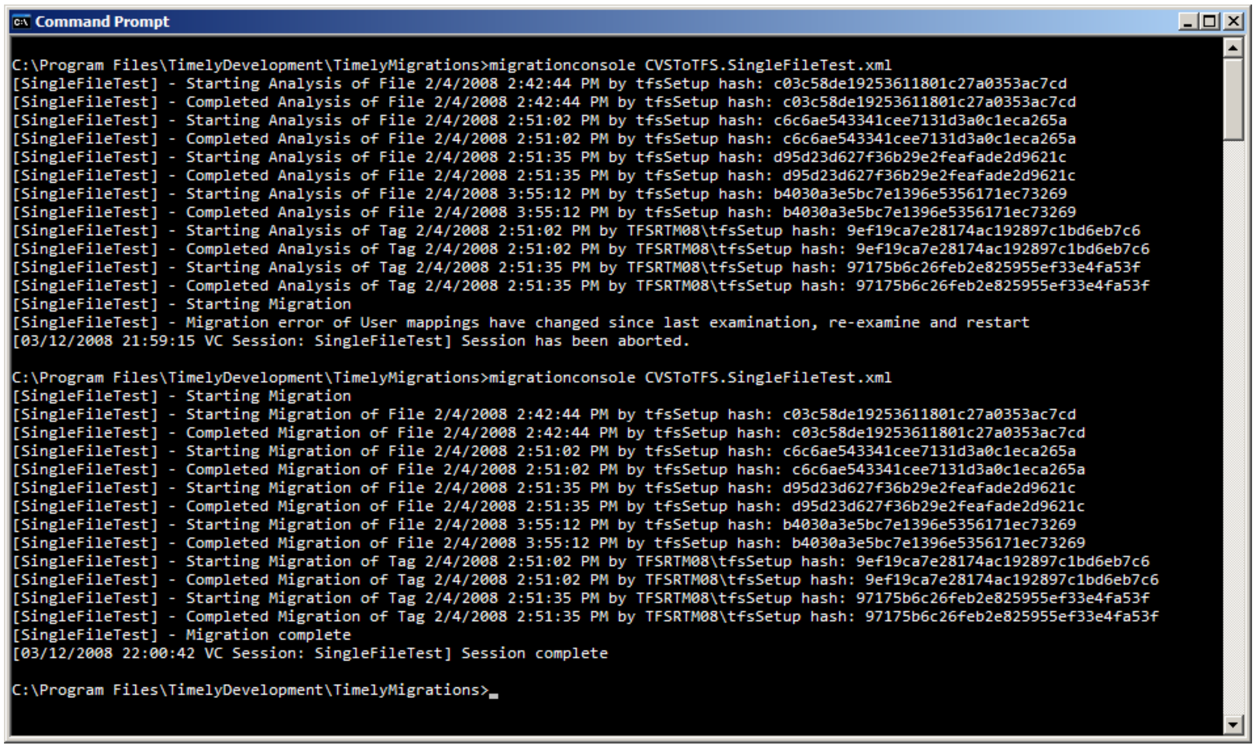
Running the Migration

A migration is performed by running a console application named MigrationConsole.exe and passing in the name of the configuration file that was created in the Configuration chapter to tell MigrationConsole.exe what you want to migrate.

As the migration runs, general status is written to the console including information about each of the file or label/tags that are being processed and whether the migration is in the analysis or migration phase. Within each phase, files will usually be processed first followed by labels/tags.

Between the analysis and migrations phases, the migration attempts to map users in the origin source database with users in the target source database. Whenever there is a discrepancy between the two, the migration will pause and force the user mappings to be reconciled before it will continue. If the user mappings are not reconciled the session will abort. The details surrounding the configuration file is discussed later in the User Mapping section.

An example of a sample migration can be seen below:



```
C:\Program Files\TimelyDevelopment\TimelyMigrations>migrationconsole CVSToTFS.SingleFileTest.xml
[SingleFileTest] - Starting Analysis of File 2/4/2008 2:42:44 PM by tfsSetup hash: c03c58de19253611801c27a0353ac7cd
[SingleFileTest] - Completed Analysis of File 2/4/2008 2:42:44 PM by tfsSetup hash: c03c58de19253611801c27a0353ac7cd
[SingleFileTest] - Starting Analysis of File 2/4/2008 2:51:02 PM by tfsSetup hash: c6c6ae543341cee7131d3a0c1eca265a
[SingleFileTest] - Completed Analysis of File 2/4/2008 2:51:02 PM by tfsSetup hash: c6c6ae543341cee7131d3a0c1eca265a
[SingleFileTest] - Starting Analysis of File 2/4/2008 2:51:35 PM by tfsSetup hash: d95d23d627f36b29e2feafade2d9621c
[SingleFileTest] - Completed Analysis of File 2/4/2008 2:51:35 PM by tfsSetup hash: d95d23d627f36b29e2feafade2d9621c
[SingleFileTest] - Starting Analysis of File 2/4/2008 3:55:12 PM by tfsSetup hash: b4030a3e5bc7e1396e5356171ec73269
[SingleFileTest] - Completed Analysis of File 2/4/2008 3:55:12 PM by tfsSetup hash: b4030a3e5bc7e1396e5356171ec73269
[SingleFileTest] - Starting Analysis of Tag 2/4/2008 2:51:02 PM by TFSRTM08\tfsSetup hash: 9ef19ca7e28174ac192897c1bd6eb7c6
[SingleFileTest] - Completed Analysis of Tag 2/4/2008 2:51:02 PM by TFSRTM08\tfsSetup hash: 9ef19ca7e28174ac192897c1bd6eb7c6
[SingleFileTest] - Starting Analysis of Tag 2/4/2008 2:51:35 PM by TFSRTM08\tfsSetup hash: 97175b6c26feb2e825955ef33e4fa53f
[SingleFileTest] - Completed Analysis of Tag 2/4/2008 2:51:35 PM by TFSRTM08\tfsSetup hash: 97175b6c26feb2e825955ef33e4fa53f
[SingleFileTest] - Starting Migration
[SingleFileTest] - Migration error of User mappings have changed since last examination, re-examine and restart
[03/12/2008 21:59:15 VC Session: SingleFileTest] Session has been aborted.

C:\Program Files\TimelyDevelopment\TimelyMigrations>migrationconsole CVSToTFS.SingleFileTest.xml
[SingleFileTest] - Starting Migration
[SingleFileTest] - Starting Migration of File 2/4/2008 2:42:44 PM by tfsSetup hash: c03c58de19253611801c27a0353ac7cd
[SingleFileTest] - Completed Migration of File 2/4/2008 2:42:44 PM by tfsSetup hash: c03c58de19253611801c27a0353ac7cd
[SingleFileTest] - Starting Migration of File 2/4/2008 2:51:02 PM by tfsSetup hash: c6c6ae543341cee7131d3a0c1eca265a
[SingleFileTest] - Completed Migration of File 2/4/2008 2:51:02 PM by tfsSetup hash: c6c6ae543341cee7131d3a0c1eca265a
[SingleFileTest] - Starting Migration of File 2/4/2008 2:51:35 PM by tfsSetup hash: d95d23d627f36b29e2feafade2d9621c
[SingleFileTest] - Completed Migration of File 2/4/2008 2:51:35 PM by tfsSetup hash: d95d23d627f36b29e2feafade2d9621c
[SingleFileTest] - Starting Migration of File 2/4/2008 3:55:12 PM by tfsSetup hash: b4030a3e5bc7e1396e5356171ec73269
[SingleFileTest] - Completed Migration of File 2/4/2008 3:55:12 PM by tfsSetup hash: b4030a3e5bc7e1396e5356171ec73269
[SingleFileTest] - Starting Migration of Tag 2/4/2008 2:51:02 PM by TFSRTM08\tfsSetup hash: 9ef19ca7e28174ac192897c1bd6eb7c6
[SingleFileTest] - Completed Migration of Tag 2/4/2008 2:51:02 PM by TFSRTM08\tfsSetup hash: 9ef19ca7e28174ac192897c1bd6eb7c6
[SingleFileTest] - Starting Migration of Tag 2/4/2008 2:51:35 PM by TFSRTM08\tfsSetup hash: 97175b6c26feb2e825955ef33e4fa53f
[SingleFileTest] - Completed Migration of Tag 2/4/2008 2:51:35 PM by TFSRTM08\tfsSetup hash: 97175b6c26feb2e825955ef33e4fa53f
[SingleFileTest] - Migration complete
[03/12/2008 22:00:42 VC Session: SingleFileTest] Session complete

C:\Program Files\TimelyDevelopment\TimelyMigrations>_
```

User Mapping

When dealing with various source control systems, it is likely that the other source control system is not tied into Active Directory or over time, employees may come and go, so the accounts that exist in your source history do not correspond to active accounts in Team Foundation Server.

To deal with these situations, a mapping file will be used to translate accounts from the origin source control system with Team Foundation Server.

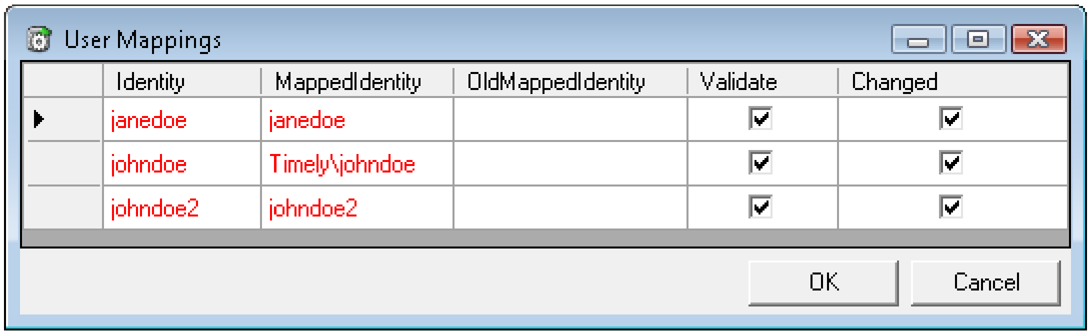
During the analysis phase, the mapping file will be built up as each change in the origin source control system is examined and new users are encountered. At the end of the analysis phase the mappings will be written out to a file named <Session Id>UserMappings.xml.

Then before the migration phase starts, the mapping file will be validated against users in Team Foundation Server. If it is the first time the migration phase has been run for the session, the migration will attempt to map users in the file to users in Team Foundation Server.

If the user is found in the system or in the active domain, the mapped user name will be updated to be the Windows account for that user. If not, the user name will be added as the mapped name so that the system will create an audit read only account to attribute the changes to.

In either case, the mapping file will have that user's entry flagged as changed. If there is an entry flagged as changed, the migration phase will display a dialog to allow you to verify that mapping makes sense and set the changed flag to false.

Let’s take the following example:

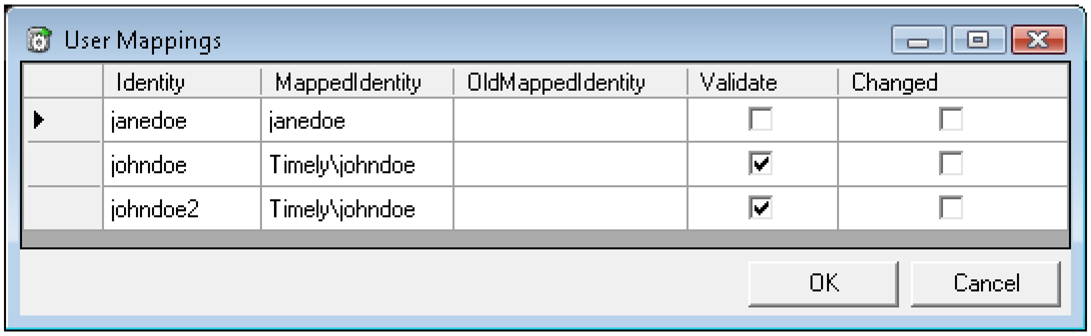


Identity	MappedIdentity	OldMappedIdentity	Validate	Changed
janedoe	janedoe		✓	✓
johndoe	Timely\johndoe		✓	✓
johndoe2	johndoe2		✓	✓

John and Jane Doe once both worked at Timely Development and contributed to code. John was forgetful and stopped being able to remember his password, so he created a second account for himself. This resulted in the johndoe login being attributed to the domain account, while johndoe2 was assigned a default audit read-only account. Since both johndoe and johndoe2 accounts were for the same person, the MappedName attribute for johndoe2 should be updated to reflect the domain account. As we are using a domain account for John, we should leave the validate flag as true so that if the account is no longer usable we will be notified by of the change and can remap the two accounts.

As for Jane, when the migration occurred, Jane no longer worked at Timely Development and a domain login for her did not exist. Because of this she was assigned an audit read-only account. Since that account will never validate to a domain account we can set the validate flag to false.

After making these changes the mapping file should look like this:



	Identity	MappedIdentity	OldMappedIdentity	Validate	Changed
▶	janedoe	janedoe		☐	☐
	johndoe	Timely\johndoe		✓	☐
	johndoe2	Timely\johndoe		✓	☐

Chapter 5: Migration Issues

Sometimes when a migration runs there are issues with the origin source database that need to be manually reconciled. Common examples are duplicate items due to case sensitive paths in CVS, StarTeam, and SVN, migrating over to TFS and occupying the same path due to TFS not supporting case sensitive paths.

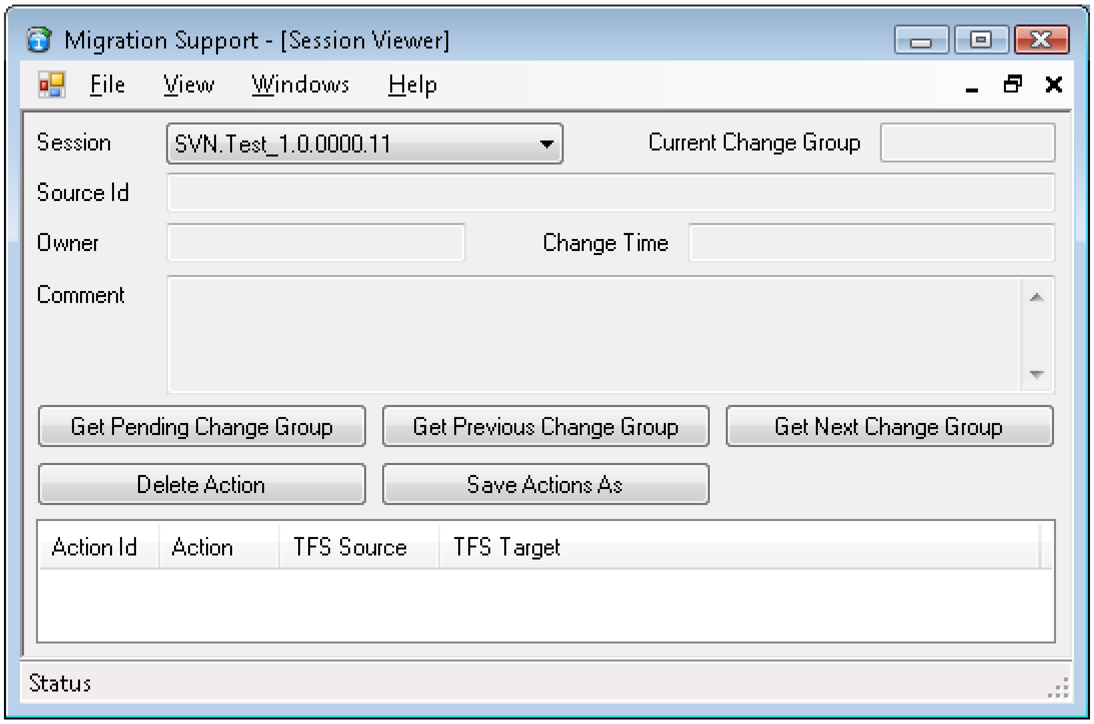
Settings like the "ItemExistsHandler" can attempt to automatically correct issues with adds and edits of duplicate items, but deletes and renames can cause problems that need to be addressed as in the following use case.

```
Add /A
Add /a
Rename /A to /B
Rename /a to /b
Delete /b
```

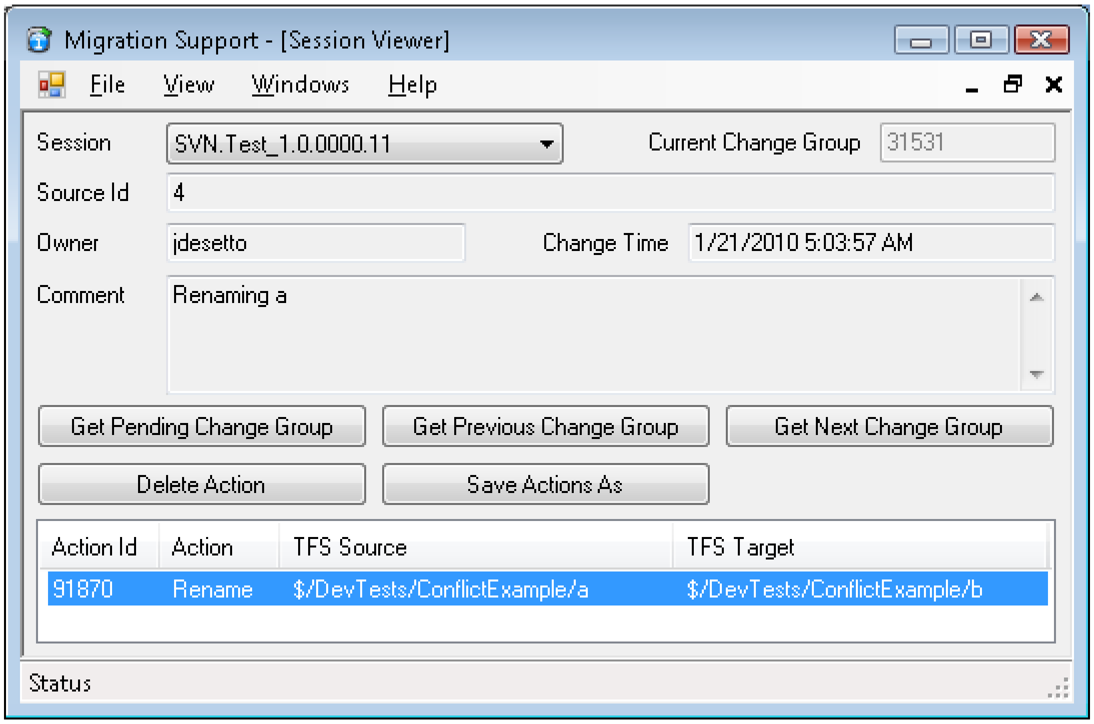
With this use case, the add of /a gets overlaid onto /A. The first rename of /A to /B succeeds, but then the rename of /a to /b will fail because only one version of A existed in TFS and it was already renamed.

From looking at the history, /b will eventually be deleted so to correct the issue the actions to rename of /a to /b can be deleted to skip past this issue as well as the delete of /b.

To delete these actions, launch the MigrationSupport.exe application and select File->New followed by SessionViewer utility. You will then be prompted to select the configuration file for the session having the issue. When the SessionViewer utility has finished loading it should look like the following:



Once that is loaded, press the "Get Pending Change Group" button to navigate to the Change Group that is having the problem.



Session: SVN.Test_1.0.0000.11
Current Change Group: 31531
Source Id: 4
Owner: jdesetto
Change Time: 1/21/2010 5:03:57 AM
Comment: Renaming a

Action Id	Action	TFS Source	TFS Target
91870	Rename	$/DevTests/ConflictExample/a	$/DevTests/ConflictExample/b

To correct the issue with the current Change Group, select the action that renames the target paths for the rename of /a to /b, and then press the “Delete Action” button to delete it.

Then to remove the action for the delete of /b, the "Get Next Change Group" button can be used to move to the next change group, where the delete action can be removed.

Chapter 6: Validating Results

Once the migration is complete it is time to validate the results. This chapter aims to provide some common examples of what to look for to see how the migration went.

Tip Comparisons

The most basic form of validation is comparing the tips of each of the branches in the source tree. There are a number of ways to perform this type of comparison. When the original version of Visual Studio 2005 version of the Team Foundation Client was shipped it was able to compare local files to versions on the server, but not entire folders. Later folder comparison functionality was added in the Team Foundation Server Power Tools add-on and made it into the Visual Studio 2008 release of the Team Foundation Client. The advantage of using this tool over alternatives like Kdiff3 or WinMerge is that you can directly checkout or add files from the difference window.

One difference between the two versions though is that the 2005 version only lets you compare to folders that are mapped within a Team Foundation Workspace. To work around that limitation the copy of the source tree from the origin source database can be pulled and then moved to where the Team Foundation Server version is mapped. Note that you will need to perform an initial get from the Team Foundation Server to make sure the workspace is initialized properly before copying over the source tree from the origin source database.

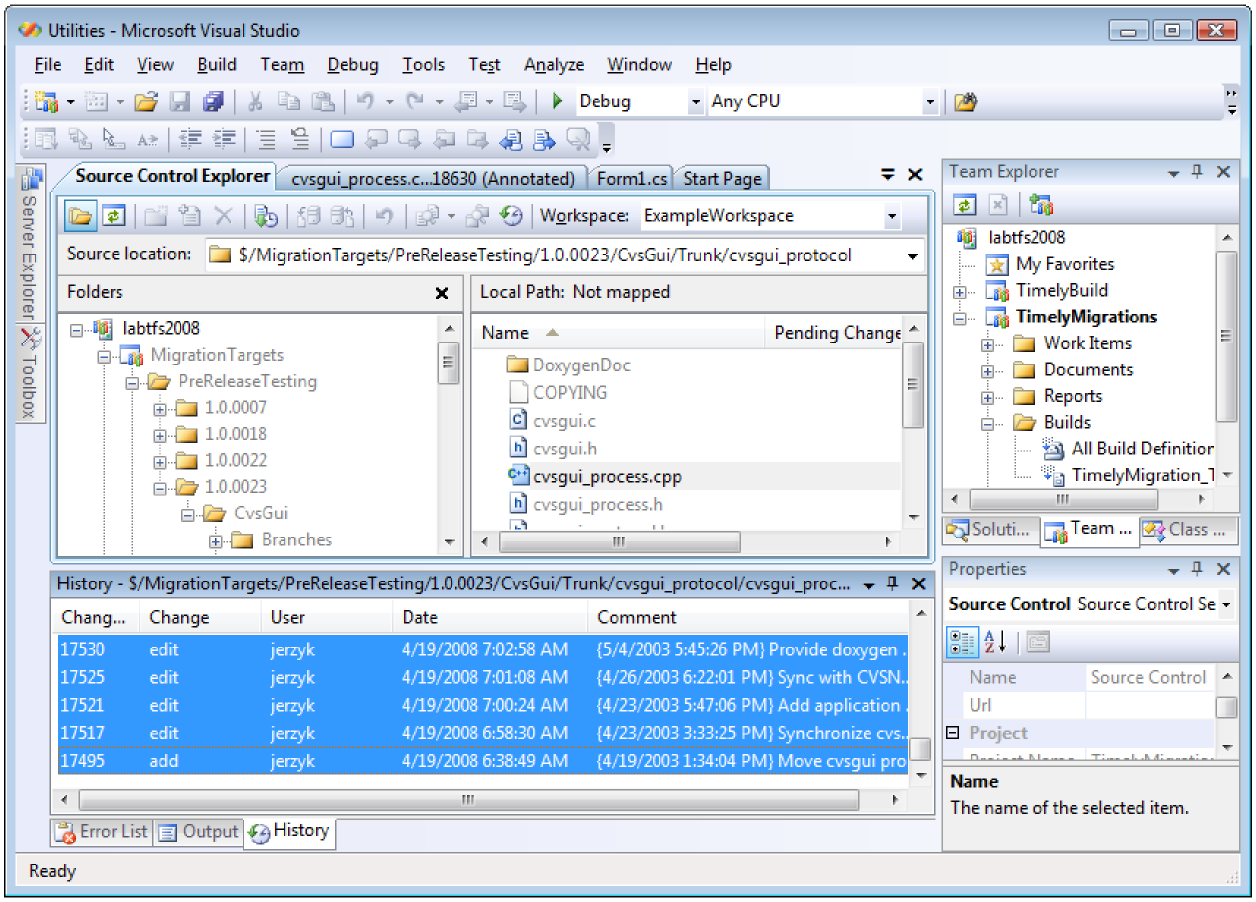
Label Comparisons

Label comparison is similar to tip comparison, but performed at certain points in time based on labels or tags that were applied to the source tree. Depending on your labeling strategy and how many builds have been performed testing every label can be impractical. To cut down on the number of comparisons, you can focus on labels that correspond to points where the source was branched.

For the labels that are not checked, it is a good idea to use the “Find Labels” dialog to check for the existence of the labels you’d expect to see and scan for any missing build numbers in label names. A missing build number may have been caused by the build failing or a problem with the origin source control system, but it is something that is easy to check for.

File Revision Count Comparisons

With TFS revisions are based off a global sequence number and not file revision numbers. If the origin source control database uses revision numbers, you can still spot check individual files to validate that revision counts match up. To do this go to the Team Foundation Client and right click on the file you want to compare and select “View History” to populate the “History” pane as shown below.


Utilities - Microsoft Visual Studio
File Edit View Build Team Debug Tools Test Analyze Window Help
Debug
Any CPU
Source Control Explorer
cvsgui_process.c...18630 (Annotated)
Form1.cs
Start Page
Team Explorer
Properties
Error List
Output
History
Ready

Workspace: ExampleWorkspace

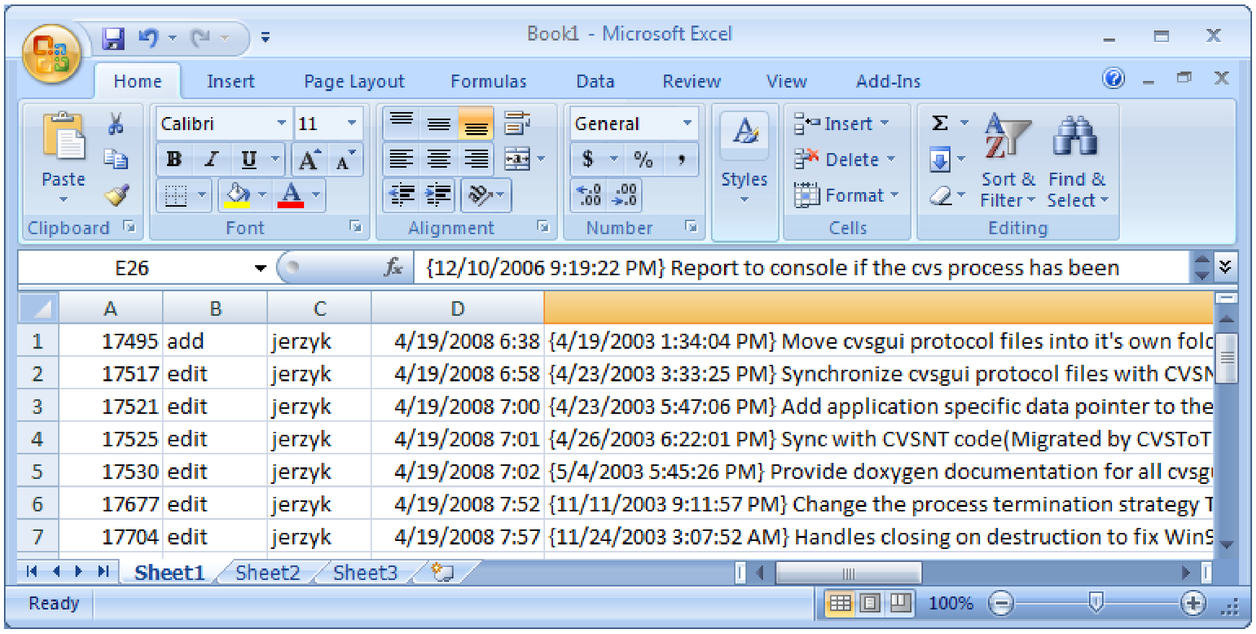
Source location: $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Trunk/cvsgui_protocol

Local Path: Not mapped

History - $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Trunk/cvsgui_protocol/cvsgui_proc...

Chang...	Change	User	Date	Comment
17530	edit	jerzyk	4/19/2008 7:02:58 AM	{5/4/2003 5:45:26 PM} Provide doxygen .
17525	edit	jerzyk	4/19/2008 7:01:08 AM	{4/26/2003 6:22:01 PM} Sync with CVSN..
17521	edit	jerzyk	4/19/2008 7:00:24 AM	{4/23/2003 5:47:06 PM} Add application .
17517	edit	jerzyk	4/19/2008 6:58:30 AM	{4/23/2003 3:33:25 PM} Synchronize cvs..
17495	add	jerzyk	4/19/2008 6:38:49 AM	{4/19/2003 1:34:04 PM} Move cvsgui pro

Once the pane is populated, you can highlight all of the entries and copy them into Excel as shown below. The changeset number will go into column A of your spreadsheet and you can resort on column A and the row numbers will become roughly equal to the revision numbers in the origin source database. Note that some operations such as adding a deleted file in CVS may translate to adding a file and then deleting the file with Team Foundation Server, so it may be valid for the revision counts to not match.



E26 {12/10/2006 9:19:22 PM} Report to console if the cvs process has been

	A	B	C	D	
1	17495	add	jerzyk	4/19/2008 6:38	{4/19/2003 1:34:04 PM} Move cvsgui protocol files into it's own folc
2	17517	edit	jerzyk	4/19/2008 6:58	{4/23/2003 3:33:25 PM} Synchronize cvsgui protocol files with CVSN
3	17521	edit	jerzyk	4/19/2008 7:00	{4/23/2003 5:47:06 PM} Add application specific data pointer to the
4	17525	edit	jerzyk	4/19/2008 7:01	{4/26/2003 6:22:01 PM} Sync with CVSNT code(Migrated by CVSToT
5	17530	edit	jerzyk	4/19/2008 7:02	{5/4/2003 5:45:26 PM} Provide doxygen documentation for all cvsgı
6	17677	edit	jerzyk	4/19/2008 7:52	{11/11/2003 9:11:57 PM} Change the process termination strategy T
7	17704	edit	jerzyk	4/19/2008 7:57	{11/24/2003 3:07:52 AM} Handles closing on destruction to fix Win9

Sheet1 Sheet2 Sheet3

Annotate

Annotate, which is similar to CVS blame, marks up a source file with notes about the revision where individual lines were introduced. This can be used as a rough estimate to see how far back history goes. In contrast to comparing file revision counts, this gives you an idea about whether or not you have the content of all revisions of the file.

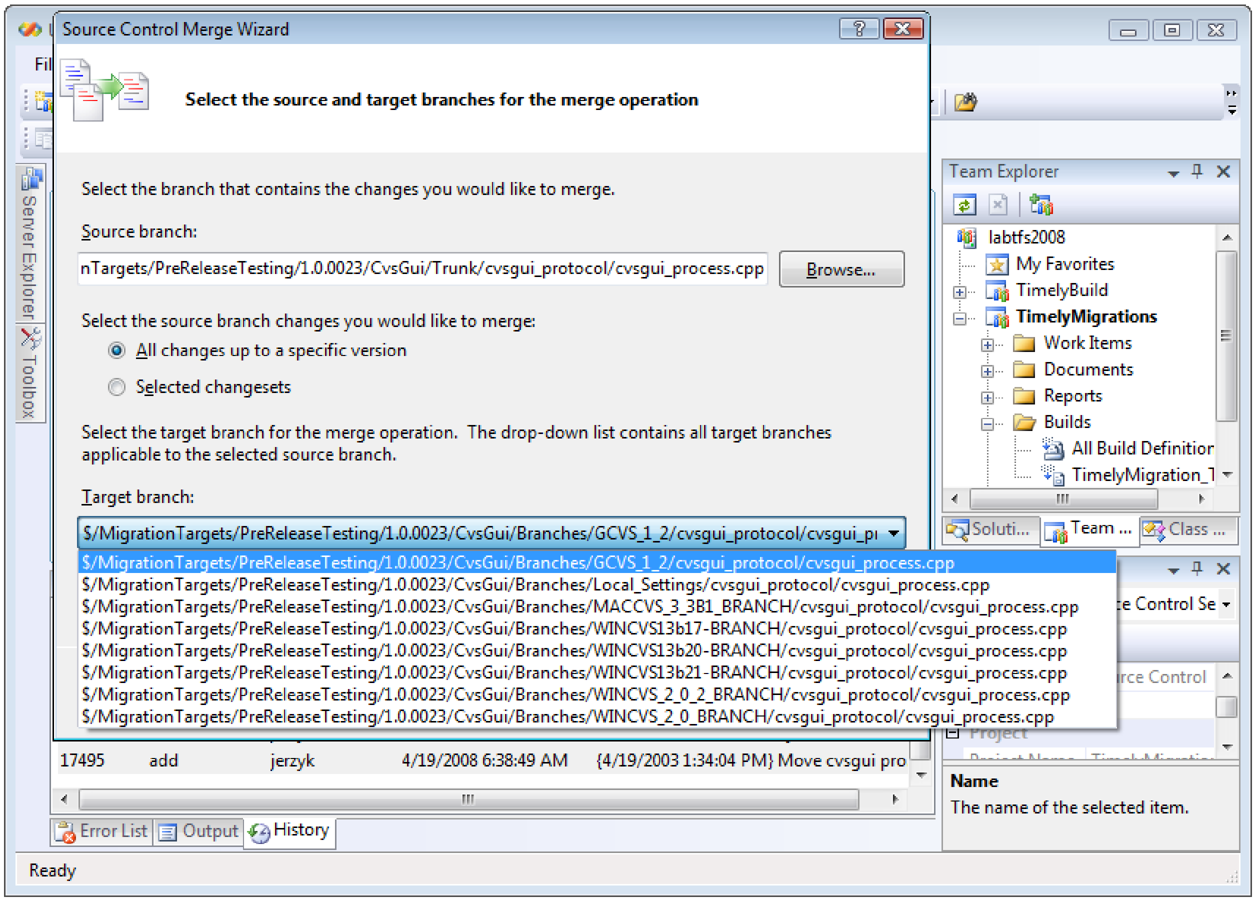
Certain sections of code such as comments and include statements at the top of files rarely change. By checking to see whether the changesets associated with these sections appear to be the among the earliest changesets for the file, you can get a feel for whether or not the content goes back to the beginning of the file. If there is a problem with retrieving the file from the origin source database, a blank file revision may be checked in for the problem revisions. This would cause the early comments to appear as if they were added in the middle or end of the file's history.

It should also be noted that Annotate is another feature that did not make the initial Visual Studio 2005 release of the Team Foundation Client but made it into the Team Foundation Server Power Tools add-on and Visual Studio 2008 release.

Branch Relationships

If the main development and release branches are being migrated, the branch relationship between the various branches should be validated that so that the build in VSTS merging functionalty can be used. This can be tested by selecting a file in the main development branch, right clicking on it and selecting "Merge" which will bring up the Source Control Merge Wizard shown below. From there expand the

"Target branch" drop down and validate that you see entries for the associated release branches that were migrated.



Select the branch that contains the changes you would like to merge.

Source branch:

nTargets/PreReleaseTesting/1.0.0023/CvsGui/Trunk/cvsgui_protocol/cvsgui_process.cpp

Select the source branch changes you would like to merge:

[✓ All changes up to a specific version]
Selected changesets

Select the target branch for the merge operation. The drop-down list contains all target branches applicable to the selected source branch.

Target branch:

$/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/GCVS_1_2/cvsgui_protocol/cvsgui_pı

- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/GCVS_1_2/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/Local_Settings/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/MACCVS_3_3B1_BRANCH/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/WINCVS13b17-BRANCH/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/WINCVS13b20-BRANCH/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/WINCVS13b21-BRANCH/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/WINCVS_2_0_2_BRANCH/cvsgui_protocol/cvsgui_process.cpp
- $/MigrationTargets/PreReleaseTesting/1.0.0023/CvsGui/Branches/WINCVS_2_0_BRANCH/cvsgui_protocol/cvsgui_process.cpp

```
17495    add    jerzyk    4/19/2008 6:38:49 AM    {4/19/2003 1:34:04 PM} Move cvsgui pro
```

Chapter 7: Advanced Configuration

For some configuration tasks the easiest way to make a configuration change may be to directly modify the configuration file without the user interface. This chapter aims to describe the structure of the configuration file so that it can be directly modified.

Common Configuration

The configuration elements described in this section are common across each of the Timely Migration modules. If individual modules override the meaning of specific settings, that will be discussed in the section for that module.

Team Foundation Server Settings

Team Foundation Server settings store information about the target source database. They are stored in the migration configuration file under a path of /Migration/Servers/Tfs as shown below. Multiple Tfs entries can be present in this section although only one may be used by each session.

```
<Migration>
  <Servers>
    <Tfs id="localhost">
      <Server>http://localhost:8080/</Server>
      <UseStoredCredentials>True</UseStoredCredentials>
    </Tfs>
  </Servers>
</Migration>
```

Descriptions for each of the fields are as follows:

Setting Name	Meaning
@id	Friendly name of the TFS server to be used in other parts of the configuration file, like the Version Control Session TFS Workspace Mapping.
Server	Fully qualified address of the TFS server.
UseStoredCredentials	Flag indicating whether or not to prompt for and store credentials for the given TFS server. If this is set to false only the credentials for the user running the migration will be used.

Version Control Session Settings

Version control session settings store various options for how data should be migrated. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Settings/Setting and use “name” and “value” attributes to store the configuration settings as shown in the example below. Only one entry with a given name should be present in a given session section.

```
<Migration>
  <VC>
    <Sessions>
      <Session id="TimelyMigration" provider="CVSProvider">
        <Settings>
          <Setting name="SkipMigration" value="true" />
          <Setting name="PrefixLabels" value="TimelyMigration" />
          <Setting name="ExclueLabels"
                   value="deadlabel1;deadlabel2" />
        </Settings>
      </Session>
    </Sessions>
  </VC>
</Migration>
```

Individual settings common between modules are described here.

Version Control Session Source Mappings

Version control session source mappings store the associate of what paths should be migrated between the source and target source control systems. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Mappings/Mapping as shown in the example below. Multiple mapping entries can be present for each of the paths that you want to migrate although the src attribute should be unique across all Mapping elements.

```
<Mappings>
  <Mapping src="$/TimelyMigration/Trunk"
           tgt="$/MigrationTargets/TimelyMigration/Trunk" />
  <Mapping src="$/TimelyMigration/Branches"
           tgt="$/MigrationTargets/TimelyMigration/Branches" />
  <Mapping src="$/TimelyMigration/Branches/DeadBranch"
           tgt="$/MigrationTargets/TimelyMigration/Branches/DeadBranch"
           cloak="true" />
</Mappings>
```

Descriptions for each of the fields are as follows:

Setting Name	Meaning
src	This setting is used to identify the folder in the origin source control database that you want to migrate. This folder should be unique. Path conventions in this setting should follow naming conventions for the origin source control system.
tgt	This setting is used to identify the folder in TFS where you want the file in the src folder migrated to. This folder does not need to be unique if you want multiple folders to map to the same destination folder. Path conventions in this setting should follow TFS naming conventions.
cloak	This setting is used to indicate whether or not you want to migrate this folder or leave it out. This can be used to prune dead branches that you do not want migrated. It can also be used to ignore individual folders that may contain content that you no longer want in your source tree such as SDK installs, sample databases or documentation.

Version Control Session TFS Workspace Mapping

The version control session TFS workspace mapping stores information about the TFS workspace that will be used for the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Tfs as shown in the example below. There may only be one of these sections per configuration file.

```
<Tfs server="localhost">
  <Workspace>Migrations</Workspace>
  <WorkspaceRoot>C:\MigrationTemp\TFS</WorkspaceRoot>
</Tfs>
```

Descriptions for each of the fields are as follows:

Setting Name	Meaning
@server	Friendly name of the TFS server to be used in other parts of the configuration file, like the Team Foundation Server Settings.
Workspace	This setting is used to identify the TFS workspace to be used with the migration. Files in this workspace will be repeatedly cleared out and updated so it is recommended that a dedicated workspace for the migration is used.
WorkspaceRoot	This setting is used to identify the folder that the workspace will be mapped to on disk. Files in this folder will be repeatedly cleared out and updated so it is recommended that a dedicated folder for the migration is used.

Version Control Session Source

The version control session source setting stores information about the origin source database that will be used in the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Source as shown in the example below. There may only be one of these sections per configuration file.

```
<Source>
  <CVS server="CVS">
    <CVSROOT>:local:\Dev\Test\TestCVSDB</CVSROOT>
    <Module>TimelyMigration</Module>
    <WorkingDirectory>C:\MigrationTemp\CVS\TimelyMigration</WorkingDirectory>
    <DownloadCacheDirectory>C:\MigrationTemp\CVS\TimelyMigrationCache
    </DownloadCacheDirectory>
  </CVS>
</Source>
```

Most of these fields will be specific to the Timely Migration module that is being used. Descriptions for those settings can be found in the Advanced Configuration section specific to that module.

Descriptions for each of the common fields are as follows:

Setting Name	**Meaning**
WorkingDirectory	This setting stores the name of a directory to be used to retrieve files from the origin source control database. The migration will overwrite all files in this directory so it should be a directory dedicated to the migration.
DownloadCacheDirectory	This setting stores the name of a directory to be used to cache individual files revisions. If repeated migrations are run, this setting can speed up the migration greatly if the database is a remote database. Additionally if there are problems with individual file revisions on the server, the file in the cache can be updated to hold a correct revision.

Staging Database

The staging database setting stores information about the SQL database to be use store session state associated with each migration. It is stored in the migration configuration file under a path of /Migration/Sql as shown in the example below. There may only be one of these sections per configuration file. In order to be able to connect to the database, it must be configured to accept local and remote connections using both TCP/IP and named pipes.

```
<Migration>
  <Sql>
    <ConnectionString>server=.\SQLExpress;Integrated Security=SSPI
    </ConnectionString>
    <Database>TfsMigration</Database>
    <Server>localhost</Server>
  </Sql>
</Migration>
```

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
ConnectionString	This setting is used to identify the SQL connection string to be used when connecting to the staging database. Take care to make sure that the correct SQL Server instance is used.
Database	This setting is used to identify the individual database to be used.
Server	This setting is used to identify the server address to connect to.

CVSToTFS Configuration

The configuration elements described in this section are specific to the CVSToTFS module of the Timely Migration suite.

Version Control Session Settings

Version control session settings store various options for how data should be migrated. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Settings/Setting and use “name” and “value” attributes to store the configuration settings as shown in the example below. Only one entry with a given name should be present in a given session section.

```
<Migration>
  <VC>
    <Sessions>
      <Session id="TimelyMigration" provider="CVSProvider">
        <Settings>
          <Setting name="CachedChangeLog"
                   value="C:\MigrationTemp\CVS\TimelyMigration\ChangeLog" />
          <Setting name="SaveParsedChangeLog" value="false" />
          <Setting name="SkipCVSInitialize" value="false" />
        </Settings>
      </Session>
    </Sessions>
  </VC>
</Migration>
```

Individual settings specific to CVSToTFS are described here.

Version Control Session Source Mappings

Version control session source mappings store the associate of what paths should be migrated between the source and target source control systems. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Mappings/Mapping as shown in the example below. Multiple mapping entries can be present for each of the paths that you want to migrate although the src attribute should be unique across all Mapping elements.

```
<Mappings>
  <Mapping src=":/" tgt="$/MigrationTargets/TimelyMigration/Trunk" />
  <Mapping src="Task_Client_A_Features:/"
           tgt="$/MigrationTargets/TimelyMigration/TaskBranches/ClientA"
           cloak="false" />
  <Mapping src="Task_Client_B_Features_Dead:/"
           tgt="$/MigrationTargets/TimelyMigration/TaskBranches/ClientB"
           cloak="true" />
  <Mapping src="*:/"
           tgt="$/MigrationTargets/TimelyMigration/Releases/*" />
</Mappings>
```

Descriptions for each of the fields are as follows:

Setting Name	Meaning
src	This setting is used to identify the folder in the CVS database that you want to migrate. This folder should be unique. With the CVSToTFS module, the src attribute will be used specify which branch you want to migrate. The path identified in src should be prefixed with "<Branch Name>:" to indicate the name of the branch tag that should be used as the basis for this branch. A "*" may be used as a wildcard indicator to specify moving all branches.
tgt	This setting is used to identify the folder in TFS where you want the file in the src folder migrated to. This folder does not need to be unique if you want multiple folders to map to the same destination folder. If a wildcard indicator is being used in the src attribute, a "*" must be present in the tgt attribute. This "*" will be substituted for the branch tag name during the migration.
cloak	This setting is used to indicate whether or not you want to migrate this folder or leave it out. This can be used to prune dead branches that you do not want migrated. It can also be used to ignore individual folders that may contain content that you no longer want in your source tree such as SDK installs, sample databases or documentation.

Version Control Session Source

The version control session source setting stores information about the origin source database that will be used in the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Source/CVS as shown in the example below. There may only be one of these sections per configuration file.

```
<Source>
  <CVS server="cvsgui">
    <CVSROOT>:pserver:anonymous:@cvsgui.cvs.sourceforge.net:/cvsroot/cvsgui
    </CVSROOT>
    <Module>cvsgui</Module>
    <WorkingDirectory>C:\MigrationTemp\CVS\cvsgui</WorkingDirectory>
    <DownloadCacheDirectory>C:\MigrationTemp\CVS\cvsguicache
    </DownloadCacheDirectory>
  </CVS>
</Source>
```

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
CVSROOT	This setting stores the CVSROOT string that CVS and WinCVS use to locate the CVS database and help determine what type of authentication, user name, and password to use.
Module	This setting stores the name of the CVS module that you want to migrate source from.

StarTeamToTFS Configuration

The configuration elements described in this section are specific to the StarTeamToTFS module of the Timely Migration suite.

Version Control Session Settings

Version control session settings store various options for how data should be migrated. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Settings/Setting and use “name” and “value” attributes to store the configuration settings as shown in the example below. Only one entry with a given name should be present in a given session section.

```
<Migration>
  <VC>
    <Sessions>
      <Session id="StarDraw" provider="StarTeamVCProvider">
        <Settings>
          <Setting name="LabelDiscoveryMode" value="RecycleBin" />
        </Settings>
      </Session>
    </Sessions>
  </VC>
</Migration>
```

Individual settings specific to StarTeamToTFS are described here.

Version Control Session Source Mappings

Version control session source mappings store the associate of what paths should be migrated between the source and target source control systems. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Mappings/Mapping as shown in the example below. Multiple mapping entries can be present for each of the paths that you want to migrate although the src attribute should be unique across all Mapping elements.

```
<Mappings>
  <Mapping src="TimelyMigration:TimelyMigration"
           tgt="$/MigrationTargets/TimelyMigration/Trunk"
           cloak="false" />
  <Mapping src="*:TimelyMigration"
           tgt="$/MigrationTargets/TimelyMigration/Branches/*"
           cloak="false" />
  <Mapping src="TimelyMigration:TimelyMigration\DeadProject"
           tgt="$/MigrationTargets/TimelyMigration/Trunk/DeadProject"
           cloak="true" />
  <Mapping src="*:TimelyMigration\DeadProject"
           tgt="$/MigrationTargets/TimelyMigration/Branches/*/DeadProject"
           cloak="true" />
  <Mapping src="DeadBranch:TimelyMigration"
           tgt="$/MigrationTargets/TimelyMigration/Branches/DeadBranch"
           cloak="true" />
</Mappings>
```

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
src	This setting is used to identify the folder in the StarTeam database that you want to migrate. This folder should be unique. With the StarTeamToTFS module, the src attribute will be used specify which view you want to migrate. The path identified in src should be prefixed with "<View Name>:" to indicate the name of the view that should be used as the basis for this branch. A "*" may be used as a wildcard indicator to specify moving all branches. Note that for StarTeam every src field should have a view name specified.
tgt	This setting is used to identify the folder in TFS where you want the file in the src folder migrated to. This folder does not need to be unique if you want multiple folders to map to the same destination folder. If a wildcard indicator is being used in the src attribute, a "*" must be present in the tgt attribute. This "*" will be substituted for the view name during the migration.
cloak	This setting is used to indicate whether or not you want to migrate this folder or leave it out. This can be used to prune dead branches that you do not want migrated. It can also be used to ignore individual folders that may contain content that you no longer want in your source tree such as SDK installs, sample databases or documentation.

Version Control Session Source

The version control session source setting stores information about the origin source database that will be used in the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Source/CVS as shown in the example below. There may only be one of these sections per configuration file.

```
<Source>
  <StarTeam server="StarTeam">
    <Server>StarTeam</Server>
    <Port>49201</Port>
    <Project>TimelyMigration</Project>
    <WorkingDirectory>C:\MigrationTemp\StarTeam\TimelyMigration
    </WorkingDirectory>
    <DownloadCacheDirectory>C:\MigrationTemp\StarTeam\TimelyMigrationCache
    </DownloadCacheDirectory>
    <User>Administrator</User>
    <Password>Administrator</Password>
  </StarTeam>
</Source>
```

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
Server	This setting stores the name of the StarTeam server.
Port	This setting stores the port that the StarTeam server is listening on.
Project	This setting stores the name of the StarTeam project that you want to migrate source from.
Username	Name of the user that should be used to connect to the StarTeam server.
Password	Password for the user that should be used to connect to the StarTeam server.

SVNToTFS Configuration

The configuration elements described in this section are specific to the SVNToTFS module of the Timely Migration suite.

Version Control Session Settings

Version control session settings store various options for how data should be migrated. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Settings/Setting and use “name” and “value” attributes to store the configuration settings as shown in the example below. Only one entry with a given name should be present in a given session section.

```
<Migration>
  <VC>
    <Sessions>
      <Session id="TimelyMigration" provider="SVNVCProvider">
        <Settings>
          <Setting name="TagFolders" value="/tags;/tags/release1" />
          <Setting name="IgnoreKeywordSubstitution" value="false" />
          <Setting name="StartingChangeset" value="0" />
          <Setting name="EndingChangeset" value="0" />
        </Settings>
      </Session>
    </Sessions>
  </VC>
</Migration>
```

Individual settings specific to SVNToTFS are described here.

Version Control Session Source

The version control session source setting stores information about the origin source database that will be used in the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Source/CVS as shown in the example below. There may only be one of these sections per configuration file.

```
<Source>
  <SVN server="SVN">
    <Server>file:///Repositories/Subversion/TimelyMigration</Server>
    <User></User>
    <Password></Password>
    <WorkingDirectory>C:\MigrationTemp\SVN\TimelyMigration</WorkingDirectory>
    <DownloadCacheDirectory>C:\MigrationTemp\SVN\TimelyMigrationCache
    </DownloadCacheDirectory>
  </SVN>
</Source>
```

Descriptions for each of the fields are as follows:

Setting Name	**Meaning**
Server	This setting stores the location of the Subversion repository to connect to.
User	This setting stores the name of the user that should be used to connect to the Subversion server.
Password	This setting stores the password for the user that should be used to connect to the Subversion server.

TFSToTFS Configuration

The configuration elements described in this section are specific to the TFSToTFS module of the Timely Migration suite.

Version Control Session Settings

Version control session settings store various options for how data should be migrated. They are stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Settings/Setting and use “name” and “value” attributes to store the configuration settings as shown in the example below. Only one entry with a given name should be present in a given session section.

```
<Migration>
  <VC>
    <Sessions>
      <Session id="TimelyMigration" provider="TFSProvider">
        <Settings>
          <Setting name="StartingChangeset" value="0" />
          <Setting name="EndingChangeset" value="0" />
        </Settings>
      </Session>
    </Sessions>
  </VC>
</Migration>
```

Individual settings specific to TFSToTFS are described here.

Version Control Session Source

The version control session source setting stores information about the origin source database that will be used in the migration. It is stored in the migration configuration file under a path of /Migration/VC/Sessions/Session/Source/CVS as shown in the example below. There may only be one of these sections per configuration file.

```
<Source>
  <Tfs server="localhost">
    <WorkingDirectory>C:\MigrationTemp\TFS\TimelyMigration</WorkingDirectory>
    <DownloadCacheDirectory>C:\MigrationTemp\TFS\TimelyMigrationCache
    </DownloadCacheDirectory>
  </SVN>
</Source>
```

Descriptions for each of the fields are as follows:

Setting Name	Meaning
@server	This setting stores the id of the TFS server that is being used as the source repository. Information about the connection for the TFS server is stored in the Team Foundation Server Settings section.

Appendix I: Acknowledgements

The Timely Migration product directly and indirectly makes use of and redistributes a number of third party software libraries.

Detailed acknowledgements can be found within the ThirdPartyLicenses directory. Each library that the Timely Migration product directly uses has a subdirectory containing the license for that library and any libraries indirectly used through that library.

High level acknowledgements are as follow:

- This product includes software developed by the SharpSvn Project (http://sharpsvn.open.collab.net/).
- This product includes software developed by CollabNet (http://www.collab.net/).
- This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
- This product includes software developed by Computing Services at Carnegie Mellon University (http://www.cmu.edu/computing/).
- This product includes software developed by the OpenSSL Project (http://www.openssl.org/).
- This product includes software developed by Sleepycat Software and Oracle (http://www.sleepycat.com/).
- This product includes software developed by Bill Menees (http://www.menees.com/).
  Copyright (c) 2002-2008 Bill Menees. All rights reserved.