|  |  |
| --- | --- |
|  | |
| Component Specification | |
|  |  |

**SCM Process**

Prepared by: Regina Ackerman

IBM

July 7, 2013

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document # | Date | Description | Author | Company |
| 0.1 |  |  | Corey Blakeborough |  |
| 0.2 | 07/21/2013 | Applied template | Stephanie Jones | IBM |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc362167433)

[1.1 Goals and Objectives 5](#_Toc362167434)

[1.2 Statement of Scope 5](#_Toc362167435)

[1.3 Inter-process Interfaces 5](#_Toc362167436)

[1.3.1 Change and Defect Management 6](#_Toc362167437)

[1.4 SCM Rules / Policy 6](#_Toc362167438)

[1.4.1 Full Software Configuration Management 6](#_Toc362167439)

[1.4.1.1 Limit Access to CIUs under Full Software Configuration Management 6](#_Toc362167440)

[1.4.2 Managed and Controlled 6](#_Toc362167441)

[1.4.3 SCM Reviews 7](#_Toc362167442)

[1.4.4 SCM Roles 7](#_Toc362167443)

[1.5 Terminology 8](#_Toc362167444)

[2 Tools 10](#_Toc362167445)

[2.1 Installation Types 10](#_Toc362167446)

[2.1.1 Visual Studio 2010 SP1 Team Foundation Server 2012 Compatibility GDR 10](#_Toc362167447)

[2.1.2 Team Foundation Server 2012 10](#_Toc362167448)

[2.1.3 Team Foundation Server 2012 Power Tools 11](#_Toc362167449)

[2.1.4 Standalone Website (View Only) 11](#_Toc362167450)

[2.2 Version Control Overview 11](#_Toc362167451)

[2.2.1 Changesets 11](#_Toc362167452)

[2.2.2 Labels 12](#_Toc362167453)

[2.2.3 Workspaces 12](#_Toc362167454)

[2.2.3.1 Cloaking 12](#_Toc362167455)

[2.3 General Processes 12](#_Toc362167456)

[2.3.1 Retrieving Files 12](#_Toc362167457)

[2.3.2 Comparing Files 13](#_Toc362167458)

[2.3.3 Checking Out 13](#_Toc362167459)

[2.3.3.1 Locking 13](#_Toc362167460)

[2.3.4 File Management 13](#_Toc362167461)

[2.3.4.1 Adding Files 13](#_Toc362167462)

[2.3.4.2 Renaming Files 14](#_Toc362167463)

[2.3.4.3 Moving Files 14](#_Toc362167464)

[2.3.5 Pending Changes 14](#_Toc362167465)

[2.3.5.1 Checking In 14](#_Toc362167466)

[2.3.5.2 Conflict Management 14](#_Toc362167467)

[2.3.5.3 Undo Pending Changes 15](#_Toc362167468)

[2.3.5.4 Shelving 15](#_Toc362167469)

[2.3.6 View History 15](#_Toc362167470)

[2.3.6.1 Changesets View 15](#_Toc362167471)

[2.3.6.2 Labels View 16](#_Toc362167472)

[2.3.6.3 Changeset Tracking 16](#_Toc362167473)

[2.3.7 View Annotation 16](#_Toc362167474)

[2.3.8 Branching 16](#_Toc362167475)

[2.3.9 Binding Solutions to Source Control 16](#_Toc362167476)

[2.4 Policies 17](#_Toc362167477)

[2.4.1 Comments 17](#_Toc362167478)

[2.4.2 Labeling 17](#_Toc362167479)

[2.4.3 Naming Conventions 17](#_Toc362167480)

[2.4.4 Building Solutions 17](#_Toc362167481)

[2.4.5 Approved Code 17](#_Toc362167482)

[2.5 Security 17](#_Toc362167483)

[3 Bibliography 18](#_Toc362167484)

# Introduction

## Goals and Objectives

The purpose of Software Configuration Management (SCM) is to establish and maintain the integrity of the Configuration Items (CI) and associated Configuration Item Units (CIU) throughout their life cycle. This document describes the organization’s standard Software Configuration Management process (SCM Process) for software work products.

SCM provides visibility into the status of the evolving software product. SCM answers the following questions: Who (made the changes?), What (changes were made?), When (were the changes made?) and Why (were the changes made?).

## Statement of Scope

SCM shall be implemented for all IBM work products supported by IBM Global Business Services on the IBM PGE ED-GIS Project. This applies to all Enhancement and Maintenance development activities.

The SCM Process applies to software baselines, project and application documents. All components supported by the IBM PGE ED-GIS project team and its Subcontractors will follow this SCM Process.

## Inter-process Interfaces

The SCM Process interfaces with other processes associated with configuration management and controlling work products:

* The Requirements Management Process describes how software requirements are defined, managed and traced throughout the software development lifecycle. (Future)
* The Project Management Process describes how software project activities including software configuration management activities are planned and tracked. (Future)
* The Software Quality Assurance Process describes how reviews are conducted during the software development lifecycle to assess conformance to the SCM Process. (Future)
* The Release Management Process describes how the Release Management team plans, schedules and manages software releases, and facilitates the final Go/No-Go decision before approving the promotion of software releases into production. (Future)
* The Peer Review Process describes the types of Peer Reviews and documents the criteria for conducting each of them.
* The Change Control Process describes the process to track and manage changes to the project’s requirements. (Future)
* Change Control Board / Executive Change Review Board Process describes the process used in identifying, documenting application changes, categorizing the risks associated with the change and reporting the information to the Executive Change Review Board. (Future)

### Change and Defect Management

The SCM Process interlocks directly with the Change Control, and Peer Review Processes to ensure all changes are managed and defects documented and managed to closure.

Software Configuration Management – to track and manage changes to software code and documents used for enhancement and maintenance activities.

Change Control Process - to track and manage changes to the project’s requirements.

Peer Review Process – to identify, track and manage defects found in reviews.

## SCM Rules / Policy

Software Configuration Management (SCM) helps in establishing and maintaining the integrity of the Configuration Items (CI) (e.g. code, requirements) and associated Configuration Item Units (CIU) throughout their life cycle. SCM also ensures that an instance of a configuration item that has been placed under SCM control can be recreated for the current and two previous baselines.

### Full Software Configuration Management

Full SCM includes stringent control of CIUs through check-in and check-out of CI/CIUs, approval for baseline changes and approval for promotion to production. Full SCM document controls include the need to maintain document change history and version control.

A baseline for an application is a set of all of the application’s configuration items and corresponding configuration item units at a given point in time required to create the application in a given environment. All baselines must be identified in the SCM Plan.

Full SCM controlled CIs and CIUs shall be maintained by retaining, at a minimum, the capability to recreate the current software baseline plus two previous software baselines.

All software code and certain requirements documents must be kept under Full SCM.

#### Limit Access to CIUs under Full Software Configuration Management

It is a business imperative that SCM Tools and manual SCM procedures (where needed) be put in place to ensure that only those IBM team members who are specifically outlined in the Application SCM Plan are authorized to access and modify CIUs. Access controls must be utilized to limit control to those individuals specifically outlined in the Application SCM Plan. CIU access controls should be as specific as possible so that an individual is permitted to access and update only those CIUS they are authorized to update.

### Managed and Controlled

Some CIs and CIUs do not require full SCM control procedures to maintain product integrity. Documents that are not retained under Full SCM must be managed and controlled by including a document version number, date and history of changes. The version number and date are incremented each time the document is changed. Draft versions and related history must also be tracked to assure product integrity.

### SCM Reviews

SCM reviews must minimally be conducted on an annual basis and prior to installing an enhancement release.

### SCM Roles

The roles associated with SCM are:

* SCM Group (comprised of SCM Administrators)
* Software Configuration Control Board(SCCB)
  + - 1. SCM Group (SCM Administrators)

Every component must have an SCM Group, although one SCM Group might serve more than one component.

The SCM Group (SCM Administrators) is responsible for the following activities:

* Creating and managing the baseline libraries
* Creating and maintaining SCM Plans
* Managing access to the baseline library
* Updating the software baselines
* Creating products from the baseline libraries
* Promoting software baselines
* Identifying work products to be placed under SCM Recording

SCM Actions

* Producing and distributing SCM reports
* Reviewing the software baselines
  + - 1. Application SCCB

The Application SCCB members include all stakeholders. The SCCB responsibilities cannot be delegated. In absence of an SCCB member, responsibilities can be delegated to another SCCB member. The delegation is to remain in effect only until such time as the SCCB member is available to resume SCCB duties.

The Application SCCB is responsible for the following activities:

* Assuring that parallel work efforts impacting the same Configuration Items (such as simultaneous or overlapping enhancement and maintenance efforts with different production rollout schedules) are properly controlled.
* Code promotion from Unit / Integration Test to each subsequent test levels and prior to hand off to production support organization.
* Evidence must also be retained evidencing that test levels were conducted.
* Application SCCB approval is also required when code is moved from one physical environment to another.
* Ensuring SCM reviews are conducted and are approved; supporting review evidence, Application SCCB approvals and other SCM compliance evidence is retained in the application process asset repository for use in the event of SQA or other internal or external reviews.
  + - 1. Separate and Distinct Members of SCM Group and Application SCCB

Members of the SCM Group cannot be members or delegate members of the Application SCCB. Similarly, members of the Application SCCB cannot also be members or delegate members of the SCM Group.

* + - 1. Release SCCB

Release SSCB reviews and authorizes baseline changes for Release Management supported software releases (e.g. major software releases), across multiple applications.

## Terminology

* **Baseline** – A specification or product that has been formally reviewed, reviewed and agreed upon, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures.
* **Configuration Item** – A Configuration Item (CI) is a class of a software work product or collection of software work products that is placed under Configuration Management, A CI may be software (e.g., code units, GUI, macros, HTML, compilers, scripts, control files, etc.) or process-related documentation (e.g., plans, standards, procedures, software requirement and design documents, etc.).
* **Configuration Item Unit** – A Configuration Item Unit (CIU) is an instance of a Configuration Item (e.g. HLD Ver. 1.0).
* **Configuration Management** – A discipline to identify and document functional and physical characteristics of Configuration Items; controlling changes to those characteristics; verifying compliance with specified requirements; and recording and reporting change processing and implementation status.
* **Promotion** – Promotion is the transfer of Configuration Items or Configuration Item Units to a higher level of control within a configuration management library system, for example from a development library to a test or production library. Promotion also includes movement of CI/CIU’s to the next planned test level.
* **Operations Documents** – Any documents, other than the defined deliverable work products that explain how to operate and maintain the software. Generally, these are provided to Managed Operations or to a local production support team.
* **Software Baseline Library** – A repository used to store Configuration Items and Configuration Item Units.
* **Version Control** – Unique identification and control used to document the creation and revision of work.

# Tools

The Team Foundation Server (TFS) software will be used for the management and tracking of CUs and CIUs for SCM purposes.

TFS items (including CIs and CIUs) are stored within the PG&E network, so access to the PG&E internal network is required for TFS work.

## Installation Types

### Visual Studio 2010 SP1 Team Foundation Server 2012 Compatibility GDR

Team Foundation Server is typically paired with an installation of Microsoft Visual Studio. In most cases, the versions should match; however, due to software requirements for development, the 2012 edition of TFS needs to be paired with the 2010 edition of Visual Studio using a “Forward Compatibility” update.

**Pre-requisites**

* Microsoft Visual Studio 2010
* Microsoft Visual Studio 2010 Service Pack 1

**Installation Steps**

* Ensure that the above prerequisites have been installed properly.
* Download the Visual Studio 2010 SP1 Team Foundation Server 2012 Compatibility GDR: <http://www.microsoft.com/en-us/download/details.aspx?id=29082>
* Complete the installer.

### Team Foundation Server 2012

If a full instance of Visual Studio is not required by the user, a standalone “shell” application can be installed containing the Team Foundation Server 2012 functionality. This application uses the framework of Visual Studio 2012, but only contains TFS functionality.

**Pre-requisites**

* (none)

**Installation Steps**

* Locate the Team Foundation Server 2012 Shell Installer.
* Complete the installer.

### Team Foundation Server 2012 Power Tools

Team Foundation Server has an optional set of tools that can integrate with the Windows shell in order to create SCM functionality that can be accomplished without opening software. These tools are ideal for team members who are not developers and do not need the full capabilities within the standalone TFS software

**Pre-requisites**

* Microsoft Team Foundation Server 2012 OR
* Microsoft Visual Studio 2012

**Installation Steps**

* Ensure that the above prerequisites have been installed properly.
* Download the Microsoft Team Foundation Server Power Tools installer: <http://www.microsoft.com/en-us/download/details.aspx?id=35775>
* Complete the installer.

### Standalone Website (View Only)

A standalone website also exists for Team Foundation Server which can be used for view-only SCM operations. No installation is necessary to view the website.

**Pre-requisites**

* Connection to PG&E Intranet

**Installation Steps**

* Visit the Team Foundation Server website for PG&E: <http://edappgistfsprd1:8080/tfs/ElectricDistCollection/EDAMGIS>
* Click on the “Source” tab.

## Version Control Overview

Team Foundation Server Version Control allows for the storage of Configuration Items (CI) and associated Configuration Item Units (CIU) in file format, along with their metadata and version history. TFS stores the latest version of each file on its server, and \keeps track of previous versions of files based on the differences between each version (reverse deltas).

Changes to TFS files are heavily monitored and tracked. Each user must check out a file in order to edit it, and check it back in to commit it back to the version control server as a new version.

### Changesets

Every time a CI or CIU is checked back into the server, the group of files being checked in is stored as a changeset. Changesets have unique ID numbers that are assigned sequentially. Changeset detail will be displayed differently depending on the context in which a changeset is viewed, because the changes stored are the deltas of each file, and not the full file.

Each changeset stores the delta for the files being checked in, the comments of the user checking the file in, the date of the check-in, the user who checked the files in, and more addition information that can be used for tracking changes.

### Labels

Labels capture a point of time for a specific group of files and/or folders. Labeling a set of items with a name and comment will associate the label with the desired version of all of the specified items (typically the latest version). Labels are useful in that they can easily be used to reflect a release version without having to look up changeset ID numbers.

### Workspaces

A Team Foundation Server workspace is stored at a user-level. A workspace stores working folder mappings. These folder mappings connect client-side folders with folders on the TFS server. The mappings indicate, for each source control folder, where the folder and files therein are stored locally and what version of the folder will be obtained from the server. Workspaces can contain any level of folder mappings. Each mapping can be recursive and apply to all folders beneath it, if desired.

Team members may have ay number of workspaces. Creating multiple workspaces may prove useful when working in multiple environments.

#### Cloaking

Within a workspace, items can be cloaked in order to exclude files or folders from being retrieved. This can be done in TFS by right-clicking a file or folder and selecting “Cloak”. Files or folders can be cloaked or uncloaked at any time.

## General Processes

### Retrieving Files

The process of retrieving a file in Team Foundation Server can be commonly referred to as a “get”. Users can get files fitting any criteria. The simplest way to retrieve files is to right-click on a tile or folder and choose one of two options:

* **Get Latest Version** finds the newest version of selected file(s), or all files and folders within selected folder(s). This is the recommended approach unless a specific alternate purpose is intended.
* **Get Specific Version** prompts the user to select or locate a specific version of the selected file(s), or all files and folders within selected folder(s). The latest version of these files will still be stored and reflected as such within the TFS server, but the local copy within the user’s workspace will reflect any version that the user chooses. The version is typically selected based on changeset ID.

Other methods for retrieving specific versions of code exist as well, including the ability to get a folder hierarchy based on one of its labels.

### Comparing Files

The “Compare” tool can be run on any text-based item on the TFS server. The tool can also be used on the read-only website version of TFS.

Users may compare a file to any previous version by selecting the file and right-clicking it, choosing “Compare”, and selecting the version to compare with. The compare tool is also visible as a right-click option when viewing modified files in a specific changeset.

In the compare tool, each version to compare occupies one side of the window. Users can easily detect via color coding what was added, removed, and modified between versions.

### Checking Out

#### Locking

By default, file checkouts are not restricted to a single user and allow multi-user checkout, meaning that users can be editing the same file at the same time. Most cases of concurrent editing will require a merge upon check-in.

This behavior is generally acceptable; however, files and/or folders can be locked in order to prevent other users from being able to modify them. Items are locked for isolation by selecting the item(s), right-clicking, and choosing “Lock”. Locks appear as pending changes and need to be checked in as if the file were modified.

Locking is an acceptable practice in many situations, but users must be diligent about its use and only keep a file locked as long as necessary.

### File Management

#### Adding Files

Files must manually be indicated as additions to the TFS server. Adding files is accomplished by right-clicking the folder (or simply somewhere inside the folder) and choosing “Add Items to Folder”. A dialog will appear asking for the location of the root folder containing the items to add.

The next screen will confirm each add process. All included files within the specified folder(s) will appear in the “Included Items” tab. Any amount of files can be excluded. For example, developers should not include “bin” or “obj” directories within the TFS server.

TFS attempts to intelligently exclude unwanted items by default, so it is recommended to check the “Excluded Items” tab and ensure that nothing is being incorrectly excluded.

The “add” operation is classified as a pending change, and these changes must be checked in after they have been made.

#### Renaming Files

Items within TFS may be renamed by selecting an individual rename and choosing the “Rename” option. The “rename” operation is classified as a pending change, and these changes must be checked in after they have been made.

#### Moving Files

Items within TFS may be moved to other folders, including folders themselves. This can be accomplished by selecting an individual item and selecting “Move”, then selecting the desired new folder. The move function is classified as a “rename” operation by TFS. This pending change can affect other changes, so it is recommended to check moved files in before continuing to make other changes.

### Pending Changes

#### Checking In

Check-ins are changeset-driven, as each individual check-in operation in TFS will create a new changeset. Users may select any number of pending changes to check in, fill out a comment indicating the details about the pending changeset, and click the “Check In” button to commit the changeset to the server as the latest version of each file.

#### Conflict Management

If conflicts are detected when checking in files or merging branches, the check-in process will be halted. Users will have to manually resolve conflicts before attempting to check in the offending file. Conflicts give multiple options for resolutions:

* **Merge** will open up the Compare Tool as described in section [2.3.2](#_Comparing_Files), with some additional features included in order to accommodate file merging. The merge tool will not be available if the conflicting file is not in a text format.

Each version (the user’s version and the server’s version) will occupy one side of the tool, and the “Result” window on the bottom indicates the result of the merge. Users can easily detect via color coding what was added, removed, and modified between versions. Checking boxes on one or both versions allows the user to keep that version’s changes. The “Result” window may also be directly modified in order to fix issues manually.

* **AutoMerge** attempts to merge the files automatically. This option will not be available if the conflicting file is not in a text format, or if the conflicts are too complex for the tool. AutoMerge may be sufficient for code changes, but a review is required. As the “Merge” option by default includes attempted AutoMerge changes and allows a code review of the merge, the Merge option is recommended over the AutoMerge option.
* **Take Server Version** resolves the conflict by ignoring the user’s local copy of the file and keeping the version stored on the server.
* **Take Client Version** resolves the conflict by ignoring the server’s copy of the file and keeping the version stored locally by the user.

Merges may not always result in successful builds, so it is important for users to ensure that code projects run correctly after a merge.

#### Undo Pending Changes

If a user needs to undo the check-out process of certain items, he/she may accomplish this by selecting any number of item(s), right-clicking, and choosing “Undo Pending Changes”. Any folders selected will automatically include all items inside. For ease of use during an impending check-in, this can be done using selected items within the “Pending Changes” window.

#### Shelving

Changesets that are checked in are committed immediately to the server as the latest version of the code. As there are policies regarding items being checked in, the need may arise for users to be able to store items on the TFS server without checking them in. This is accomplished via shelving. A group of shelved items is referred to as a shelveset. A shelveset stores items in a similar way to a changeset, but the changes (deltas) of each file are not stored as the latest version of the file; instead, a user would have to manually retrieve the shelveset to be able to have the changes therein.

Shelvesets are recommended for peer review, temporary storage, or work handoff because the latest version of the files in the server are not affected. Once a shelveset has fulfilled its purpose and is no longer needed, it should be deleted.

### View History

An item’s history may be viewed by selecting the item, right-clicking it, and selecting “History”. The resultant window displays the history of the file or folder since creation.

#### Changesets View

The default view for the history window is the Changesets view. This view shows every changeset that modified the selected file (or one of the files within the selected folder). Double-clicking on a changeset brings up a detailed view showing every file that was modified as well as more information. The compare tool is a useful companion to the Changesets view.

#### Labels View

Clicking on the “Labels” tab will show the Labels view. This view shows every label that has ever been applied to this item. Details of the label can be viewed by double-clicking the label itself. This view also allows users to pull the code from a specific label.

More information about labels is found in section [2.2.2](#_Labels).

#### Changeset Tracking

Within the Changesets view, users may select a changeset and choose the “Track Changeset” option to view the Changeset Tracking window. This window is only pertinent to items that are contained within a branch. This view shows visually the process of an item throughout each branch.

### View Annotation

Users can view the annotation of any text-based file by selecting it, right-clicking, and choosing “Annotate”. The annotation window shows the currently-pulled version of the code and annotates the side of the code based on who made each line change, what changeset it was made in, and when it was made.

### Branching

TFS supports branching code. Code branching can complicate structure, but ultimately caters to feature/team development. Each branch represents an isolated, asynchronous track that the code may take. Branches may be made from any point in the code, and may be merged at any time as well.

Like changesets, branches are stored by the changes (deltas) in the code.

Users should consult with a manager before deciding to branch configuration items stored within TFS.

### Binding Solutions to Source Control

Code managed in Microsoft Visual Studio can be “bound” to Team Foundation Server to allow additional benefits. A solution that has been bound to source control offers tighter TFS version control integration. The largest benefit is that code will automatically be checked out as changes are made to a file without explicitly needing to check the file out. Additionally, files and folders added to the solution through Visual Studio will be added to TFS as well (pending a check-in).

It is recommended that IBM-managed code be bound to source control. Opening an unbound solution in Visual Studio will prompt the user to bind the solution. The files associated with this process will need to be checked in when the binding has been completed.

## Policies

### Comments

Users must always provide a comment with each changeset or shelveset.

### Labeling

Every release of a specific product or task must be labeled.

### Naming Conventions

Files must be named intuitively according to IBM naming convention standards; however, version number and date information should not be appended to the file name as TFS manages that information within metadata. Ideally, files and folders should be named permanently, so that rename operations are infrequent or completely unnecessary.

New folders or projects should be placed in their proper directories according to the established TFS folder structure.

### Building Solutions

Code that does not build successfully must not be checked in. Use shelvesets to share incomplete code through TFS. Merging often breaks builds, so users should always attempt to build again after a merge operation has been completed prior to checking in the merge results.

### Approved Code

In situations where code approval is required, all code must be approved prior to being checked in, unless the code is in a self-managed branch. Use shelvesets to share unapproved code through TFS.

## Security

TFS uses roles for security and determines user information via Active Directory Integration. Users and/or Groups in the Active Directory are assigned to roles. These roles are given security permissions that can be applied at different levels throughout the project.

TFS Administrators and/or IT personnel may be required to change user permissions.

# Bibliography

Capability Maturity Model for Software Version 1.1*, Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, 1993, (SEI-93-TR-024), Software Engineering Institute, Carnegie Mellon University.*

Software Configuration Management, Version 2.1.C2, *Bob Gloyd, Mike Holt, Bob Kess, Carol Brenner, Leap Process Asset Library*