TEAM FOUNDATION SERVER (TFS) STANDARDS

Information Systems Branch | Economy Sector

Version 1.0 - Draft

August 22, 2017

TABLE OF CONTENTS

REVISION HISTORY

Date	Version	Author	Description
August 18, 2017	1.0	Andreas Ritzer	Initial Draft

ABBREVIATIONS

Abbreviation	Description
CMMI	Capability Maturity Model Integration
DEV	Development
DEVRC	Development Release Candidate
DVCS	Distributed Version Control System
FI	Forward Integrate
RI	Reverse Integrate
TFS	Team Foundation Server
TFVC	Team Foundation Version Control

DEFINITIONS

Term	Definition
DEVRC	The development release candidate is the release that is promoted from the development environment to the testing environment.
DVCS	A distributed version control system that uses a local repository to track and version files. Changes are shared with other team members by pushing and pulling changes through a remote, shared repository.
FI	The process of merging from a parent branch to a child branch.
GIT	A widely adopted distributed version control.
Master	The master branch represents the code base that currently exists in production.
RI	The process of merging from a child branch to a parent branch.
TFS	The Microsoft product that supports various aspects of Application Lifecycle.
TFVC	A legacy version control system used by TFS.

1. OVERVIEW

Team Foundation Server is the application used for supporting application development and maintenance in the Economy Sector. TFS provides various tools and features that are of use:

- Source code management
- Automated builds
- Release Management

The current version in use by the Economy Sector is **Team Foundation Server 2015**.

2. TFS Projects

2.1. Project Naming

TFS projects will be named according to the following standard:

<Project Abbreviation> - <Project Full Name>

For example:

HEC – Hiring Evaluation Calculator

TFS projects are created by the Economy Sector Administrator's.

2.2. TFVC / GIT

TFS provides two version control options for projects within TFS:

1. **TFVC** – A single, centralized server repository is used to track and version files. All local changes are always checked into the central server so that other team members can view the latest changes.
2. **GIT** – A distributed version control system that uses a local repository to track and version files. Changes are shared with other team members by pushing and pulling changes through a remote, shared repository.

Microsoft strongly encourages users to use the GIT version control system and have indicated that all future development efforts in TFS will focus on GIT enhancements before TFVC is considered. With that in mind **the default version control choice when creating a project in the Economy Sector TFS instance will be GIT.**

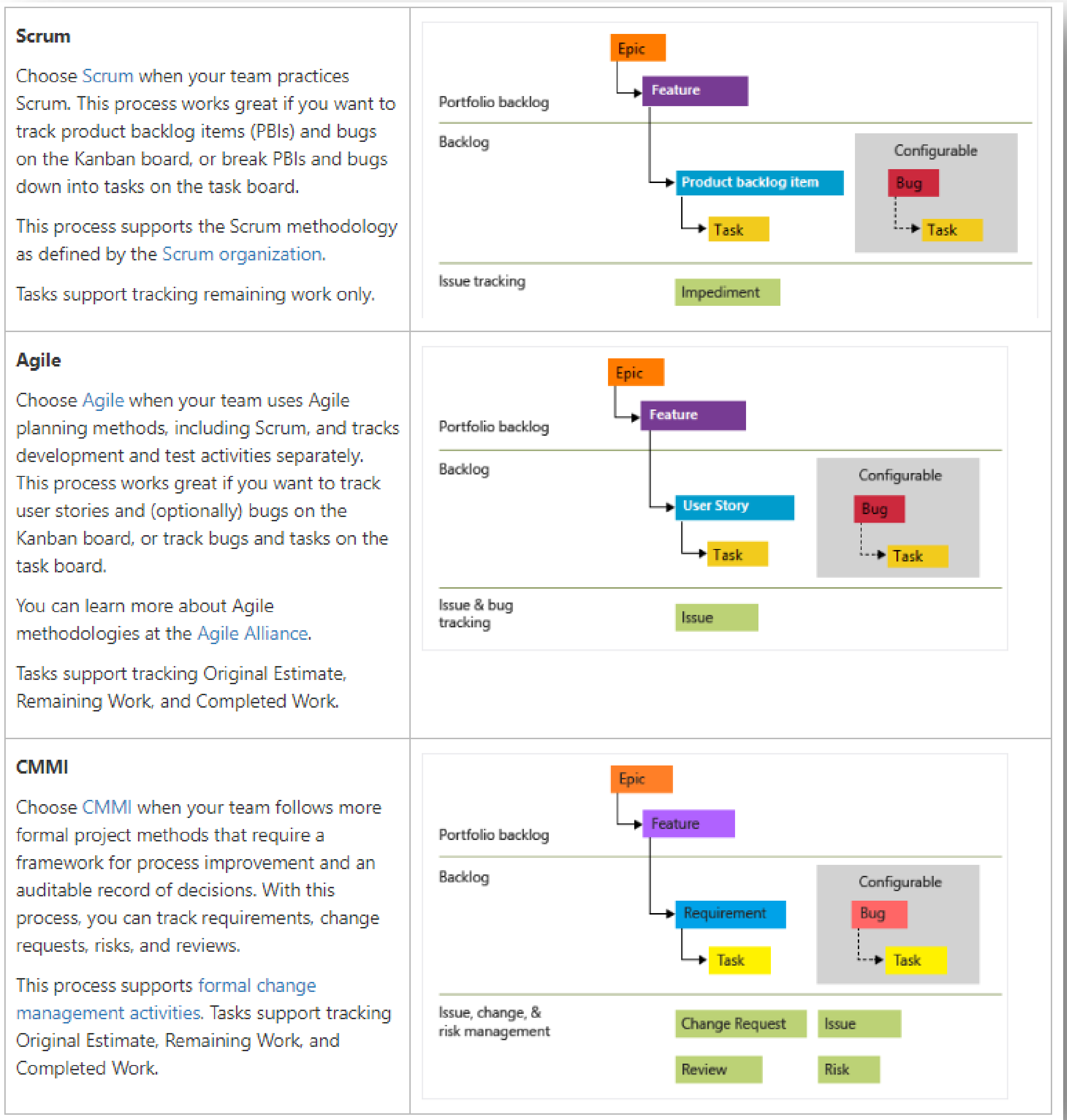
If GIT is not feasible for the project team the reasons TFVC is required must be documented and provided to the Technical Architect team for consideration/review.

2.3. Process Templates

Upon project creation, the following process templates can be chosen:

1. Scrum
2. Agile
3. CMMI

Microsoft provides the following definitions and work item flow for each process template:



Scrum

Choose Scrum when your team practices Scrum. This process works great if you want to track product backlog items (PBIs) and bugs on the Kanban board, or break PBIs and bugs down into tasks on the task board.

This process supports the Scrum methodology as defined by the Scrum organization.

Tasks support tracking remaining work only.

Portfolio backlog
Epic
Feature
Backlog
Product backlog item
Task
Configurable
Bug
Task
Issue tracking
Impediment

Agile

Choose Agile when your team uses Agile planning methods, including Scrum, and tracks development and test activities separately. This process works great if you want to track user stories and (optionally) bugs on the Kanban board, or track bugs and tasks on the task board.

You can learn more about Agile methodologies at the Agile Alliance.

Tasks support tracking Original Estimate, Remaining Work, and Completed Work.

CMMI

Choose CMMI when your team follows more formal project methods that require a framework for process improvement and an auditable record of decisions. With this process, you can track requirements, change requests, risks, and reviews.

This process supports formal change management activities. Tasks support tracking Original Estimate, Remaining Work, and Completed Work.

Figure 1 Microsoft definitions for available for TFS Process Templates

The Economy Sector supports the CMMI and Agile process templates to coincide with the project methodologies we support. The Scrum process template is **not** used.

2.3.1. Process Template Selections

For waterfall methodology projects, the CMMI process template will be used. For agile projects, the Agile process template will be used.

3. Version Control

3.1. Mandatory Folders

To ensure consistent use of the version control aspect of TFS the following folders are mandatory in each TFS project version control (code) section:

Folder	Description
docs	This folder contains project related documentation and is not specific to a particular release. Example documentation includes: • SDD – Solution Design Document • BRD – Business Requirements Document • Data Models • Developers Guides
src	This folder is the root folder for all source code related content. All project branches (refer to branching section for details) are contained within this folder.

3.2. Folder Naming

Folder names should be descriptive but should not exceed 30 characters to avoid excessive pathing lengths. Source code folders should be as short as possible to ensure the deepest folder depth does not exceed 260 characters (given this is a well-known limitation in Windows and will cause any automated builds and/or deployments to fail).

4. BRANCHING

Regardless of the version control chosen (GIT or TFVC) the Economy Sector branching standard should be followed. The standard has taken best practices from many areas of the IT industry and closely resembles a commonly used branching strategy: GitFlow.

The key branches in the strategy are the **DEV** and **MASTER** branches. These branches are perpetual while all other branches created will be transient in nature.

4.1. BRANCH TYPES

4.1.1. DEV

The Dev branch is the active branch for development and is the location that the developers share a code base.

4.1.2. Features

Feature branches are created by individual developers and are typically located on the individual's workstation and is not stored on the shared repository (GIT or TFVC). The purpose of the feature branch is to allow the developer to work on a feature, bug, enhancement, etc. in isolation without adversely affecting other users.

Feature branches can be shared with other developers to aid in peer programming. Only GIT version control supports this type of remote sharing of branches.

4.1.3. Release

Release branches are created when a consensus has been reached that all commits to the DEV branch includes all features expected for the release. Once a release branch is created no new features of changes are made to the release branch. However, bug fixes identified in QA/UAT are addressed in the release branch.

Once a release branch is created development efforts can continue for future releases on the DEV branch.

4.1.4. Master

The Master branch does not allow active development or commits to be made to it. The sole purpose of the Master branch is to contain a historical view of all releases that have been made to production. Therefore, the only commits made to Master occur when a Release branch is promoted to production forcing a forward integration of the Release branch to the Master branch. During this promotion to the Master branch a tag must be created to reflect the version number of the release promoted to production.

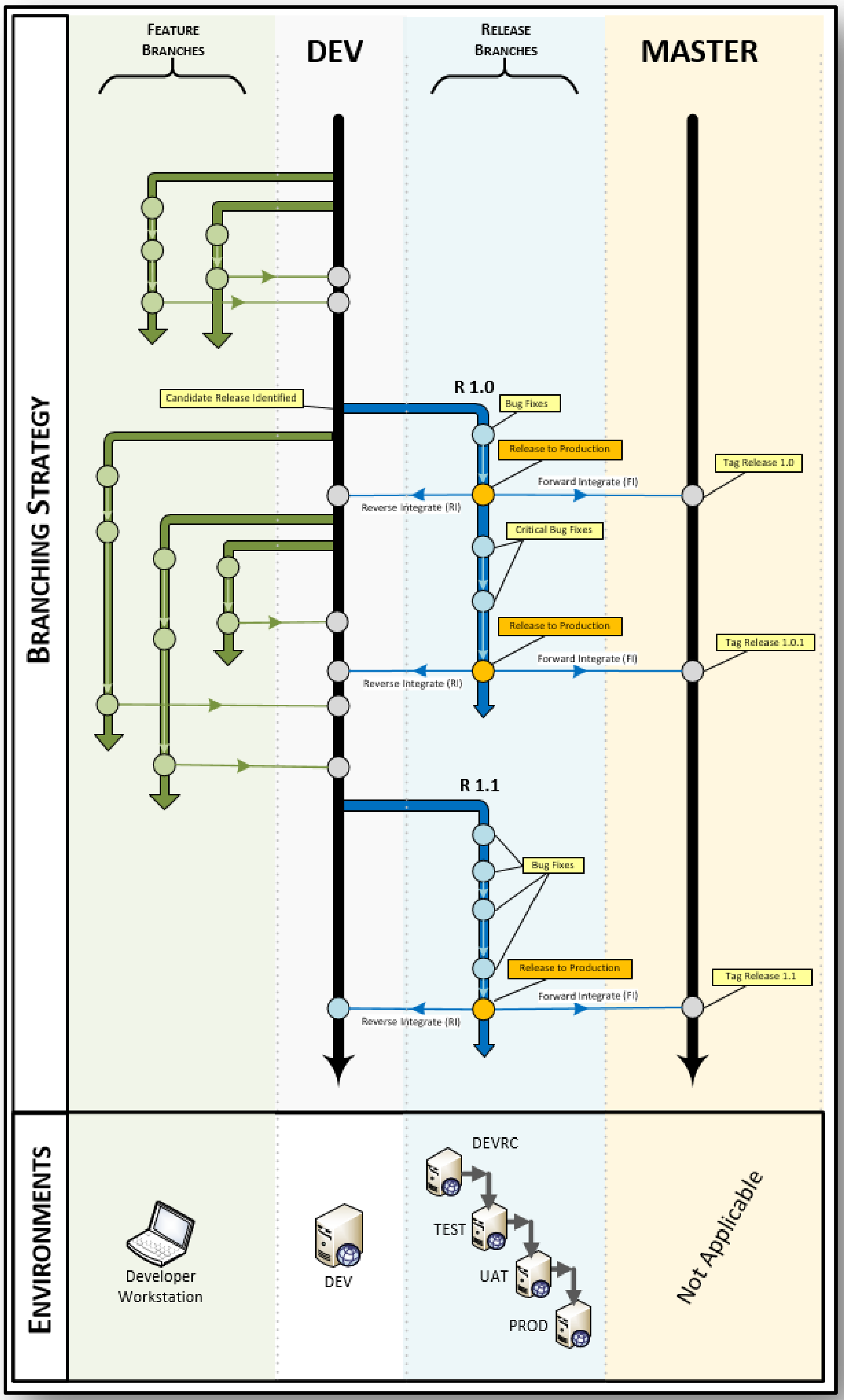


Figure 2 Economy Sector TFS Branching Strategy

5. AUTOMATION

5.1. BUILDS

TFS is used to perform automated builds. To ensure a consistent approach in the build steps the following standards must be adhered to.

5.1.1. Build Definitions

A single definition should be created for an application and be used to create the build artifact that is released to all environments (e.g. development, testing, QA, production). The name of the build definition should be self-describing in regards to the project as the name is visible when a build agent is performing the automated build. The recommended naming standard is:

*<Project Abbreviation>*_Build

For example:

HEC_Build

5.1.2. Build Steps

The following are TFS build steps that are approved for use in the Economy Sector TFS environment. If the build steps provided does not accomplish a requirement for a project a request must be made to the Economy Sector TFS administrator group for consideration.

Command	Description
Command Line	This task can be used to execute commands on the build server during the build process.
Visual Studio Build	The main task to compile application code using Visual Studio.
NuGet Installer	This task will run the NuGet command to restore all packages with the associated solution.
Visual Studio Test	The main task used to execute automated tests authored in Visual Studio.
Publish Build Artifacts	This task will publish the application artifacts and make it available for releasing into environments.
Copy Files	This will allow files to be copied from one area to another.
Delete Files	This will allow files to be deleted.

5.1.3. Build Number Format

When a build is created by TFS the build number it assigns is visible in many areas thus it is important to ensure the build number format follows a pre-defined standard to ensure team members can recognize the build.

<Application Abbreviation>-$(BuildConfiguration)-$(date:yyyyMMdd)$(rev:.r)

For example:

HEC-Release-20170402.9

5.2. Release Management

TFS is used to perform automated releases. To ensure a consistent approach in the release steps the following standards must be adhered to.

5.2.1. Release Definitions

A single definition should be created for an application and contain all the environments that the application will be released to.

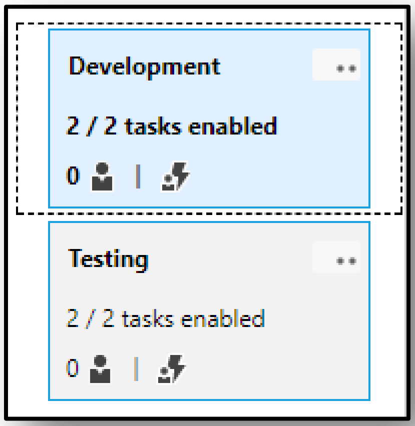


Figure 3 Multiple Environments for a Single Release Definition

and be used to create the build artifact that is released to all environments (e.g. development, testing, QA, production). The name of the build definition should be self-describing in regards to the project as the name is visible when a build agent is performing the automated build. The recommended naming standard is:

*<Project Abbreviation>*_Release

For example:

HEC_Release

5.2.2. Release Steps

The following are TFS release steps that are approved for use in the Economy Sector TFS environment. If the release steps provided does not accomplish a requirement for a project a request must be made to the Economy Sector TFS administrator group for consideration.

Command	Description
Windows Machine File Copy	This task can be used to copy build artifacts to the destination server.
WinRM – IIS Web App Deployment	This task will allow for web deployments to be performed on the destination server.
PowerShell	Provides the ability to execute PowerShell scripts on the remote server.
Copy Files	Provides the ability to copy files from remote machines.

5.2.3. Configuration

To facilitate deployment to various environments it is imperative to use configuration variables that are environment specific. Some example configuration variables to include are:

- ApplicationConnectionString
- EnvironmentTitle
- DebugMode

5.2.4. Release Number Format

When a release is created by TFS the build number it assigns is visible in many areas thus it is important to ensure the release number format follows a pre-defined standard to ensure team members can recognize the build.

<Application Abbreviation>-Release-$(rev:r)

For example:

HEC-Release-31