**ABC Source Control**

Source control allows the proper maintenance of ABC’s codebase. It provides a clear “master” version of any file, along with version control and detailed change tracking.

ABC’s chosen source control solution is Microsoft Team Foundation Server. Licenced developers interact with TFS through Visual Studio and a web interface. Non-developers have limited browser access, to raise work items for example.

**Corporate Principles / Best Practices**

* The collection is split into Team Projects corresponding to areas of work, e.g. AdViewability, CVTools etc
* ABC operates a single-branch model. Under each Team Project is a “Main” branch and code is edited there; no branch / merge operations occur
* As far as possible, spaces in directory and filenames should be avoided
* Intercapped is the preferred style for directories – e.g. MyTestDirectory
* The naming convention for scripts should reflect functionality (e.g. Sun\_iPad\_delim)
* Checked out files should be checked in as soon as possible
* When checking in a changeset, accompanying notes detailing the changes *must* be provided, and any relevant work items attached
* Changesets shouldn’t span Team Projects
* When deploying code for release, the changes should be detailed in an accompanying release notes file

**Working With Files**

There are essentially three locations involved in the process:

* The Team Foundation Server instance (ABCVSTFS) which holds the “master”
* The workspace on the developer’s PC
* The deployment target for released builds

The process of a requested change is typically:

* Get latest version of relevant file (syncs local workspace copy with server version)
* Check out file
* Open / edit / save
* Update release notes and deploy with changed file to target
* Check in file, providing technical notes and attached work item (if appropriate)

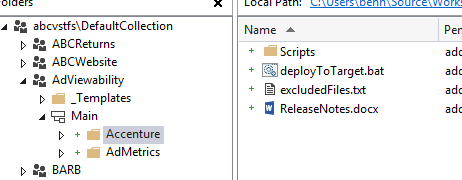
***Structure of a Team Project***

Files to be excluded from deployment

Batch file to deploy scripts

Script directory – everything below this level will be deployed

Release notes for deployment



Team Project

Admin templates

Main branch

Subdirectories split by client

Team projects should be split by client. Each client subdirectory should contain:

* A Scripts subdirectory; this folder will be copied to production during deployment
* deployToTarget.bat to initiate deployment (available under \_Templates), parameterised with the deployment path
* excludedFiles.txt to list files which shouldn’t be deployed – this includes at minimum itself and the batch file
* ReleaseNotes.docx to be updated and included with every deployment

Each Team Project is split into subdirectories by client, with a Scripts subdirectory. A read-only copy of this scripts directory should be created in a shared network location for use by customers.

A simple batch file (deployToTarget.bat) is included in the \_Templates directory of each Team Project; a version should be placed in each client directory at the same level as the Scripts folder. The file excludedFiles.txt should also be included – this is a simple list of filenames to be ignored. *Note that this is the simplest requirement. If there is a more granular split beneath the Scripts subdirectory, it may be preferred to move the administrative files to the appropriate level.*

The file deployToTarget.bat is a simple process which copies everything from the current directory downwards to a target folder. When setting up the file structure in source control, the batch file must be amended with the appropriate target:

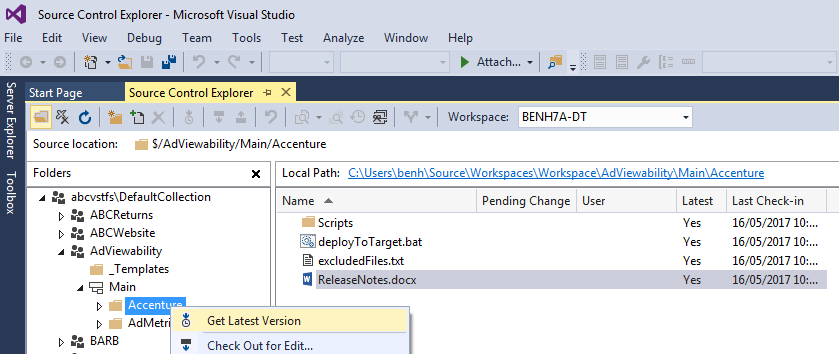


Change to relevant path

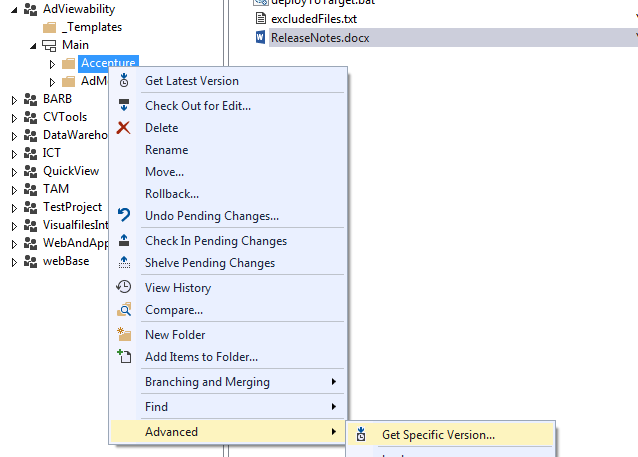
***Get version***

The first action when working with source control should always be to perform a Get operation. This ensures that the files in your local workspace are the latest versions. A Get operation can be performed at any level from an individual file all the way up to the root Team Project.

In Visual Studio, use Source Control Explorer to navigate the directory tree. Right-click on the relevant directory or file and select Get Latest Version from the context menu:



Alternatively, you can retrieve older versions of the files through Advanced > Get Specific Version:

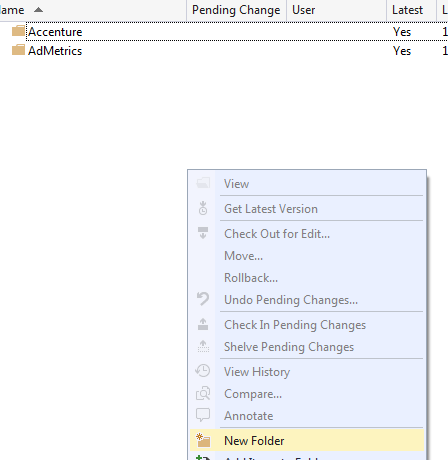


This is detailed further in the *Rolling Back* section.

***Adding files to source control***

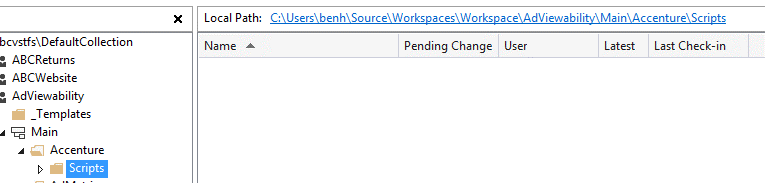
Files are added through Visual Studio. Specifically, they are added into your workspace, then pushed to TFS via a Check In.

Directories can be created by right-clicking in Source Control Explorer and selecting New Folder.

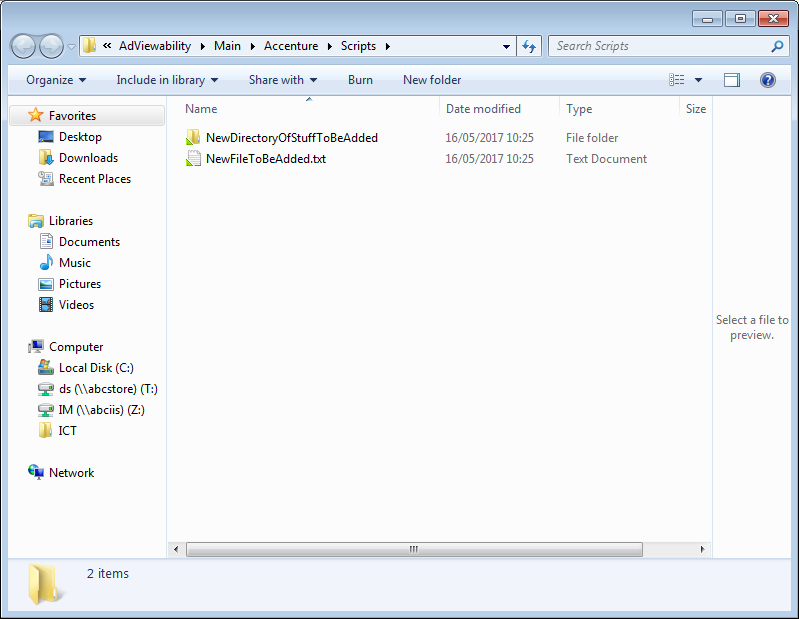


The easiest way to add files, either individually or in bulk is as follows:

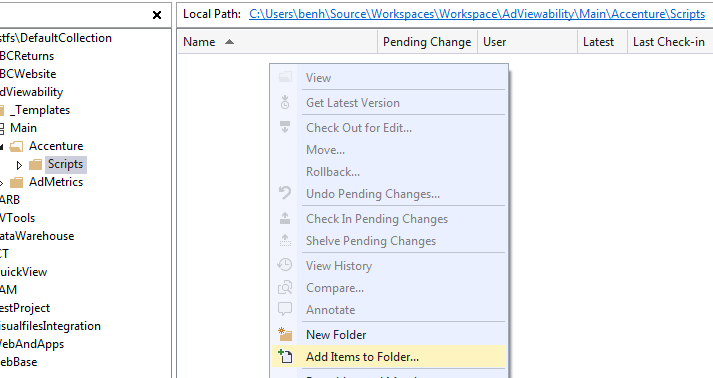
Navigate the folder tree in Source Control Explorer to the appropriate location and click local path to open Windows Explorer:



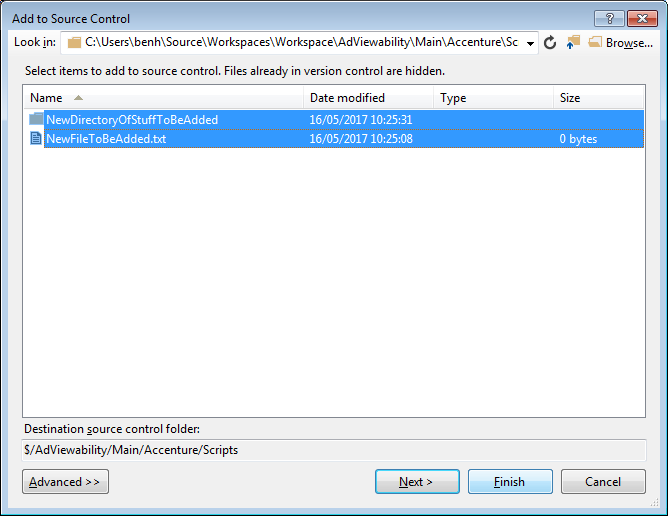
Copy / create files in this directory:



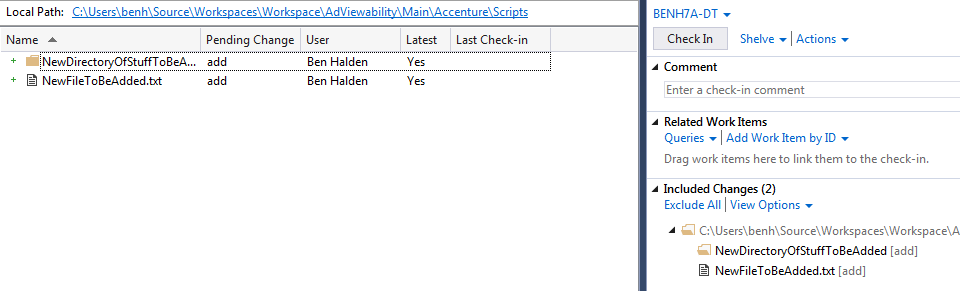
Back in Source Control Explorer, right-click and choose “Add Items to Folder” from the context menu:



Select the files to be added and press Finish:



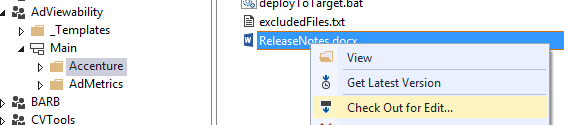
The files have been added to your local workspace, and will be included in the next Check In operation:



***Check Out***

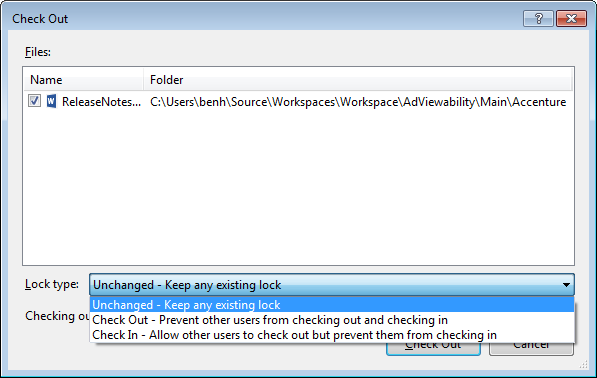
All files in your local workspace are locked and read-only. To make changes to a file, it must first be checked out. TFS is a centralised source control system, so the status of files is communicated back to the server to avoid conflicts.

To check out a file, right-click on it in Source Control Explorer and select Check Out for Edit from the context menu:



Only check out the files you wish to edit, rather than entire directories.

Choose the lock type and select Check Out;

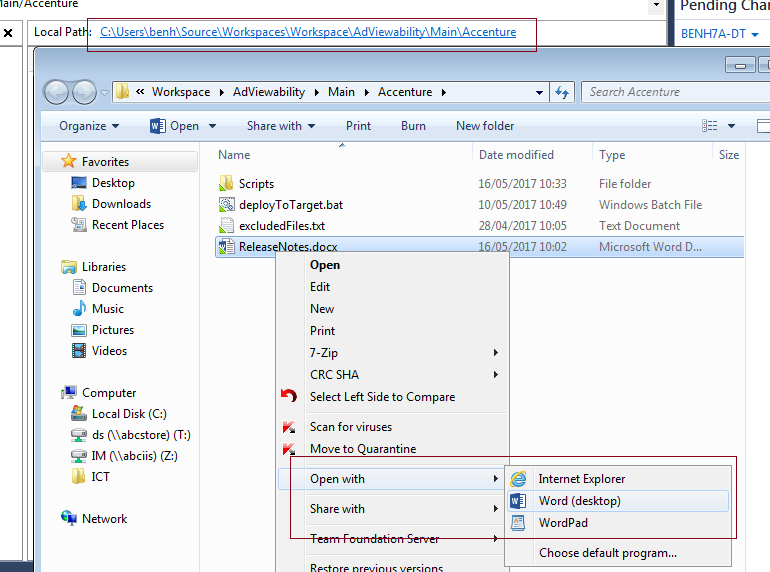


The Check Out lock type should be used here; this prevents any other developer from doing likewise, avoiding conflict issues.

***Editing***

After a file has been checked out, perform your edits. Opening the file through Source Control Explorer will, generally, open it in Visual Studio. This behaviour can be changed by overriding the default application for files by extension type (e.g. open .sql files with SSMS).

Alternatively, click Local Path to open windows explorer and open the file from there as desired.

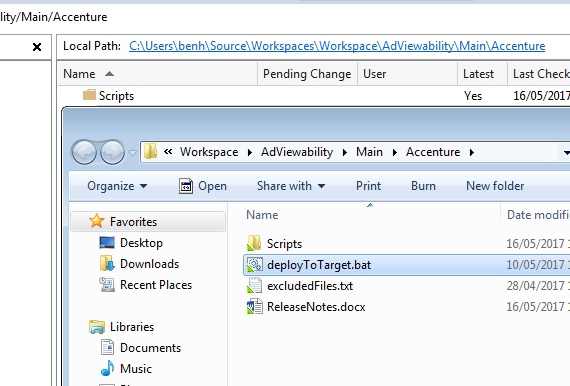


***Deployment***

Customers do not have access to Source Control, so files must be deployed to production for use by, e.g. auditors. What this process entails can vary by the nature of the files in source control, but for our purposes here, the process is as follows.

* When files are ready for deployment, they should be saved and closed
* The relevant ReleaseNotes.docx should be updated with details of the release
* Click local path in Source Control Explorer to open Windows Explorer
* Run deployToTarget.bat

Run



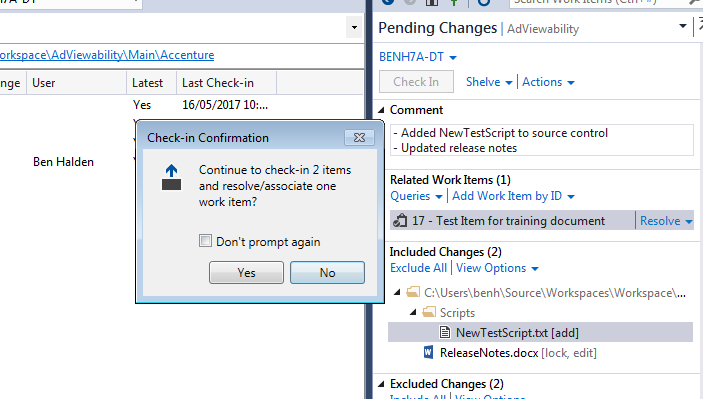
These are copied to target

***Check In***

Checked out files should be checked in as soon as possible. In Visual Studio, in the Team Explorer panel, navigate to Pending Changes, ensuring you are in the relevant Team Project:

Related work item (if appropriate)

Mandatory comments



Changes to be committed in this changeset

A preview of what will be included in the changeset is displayed here; files may be excluded at this point if desired.

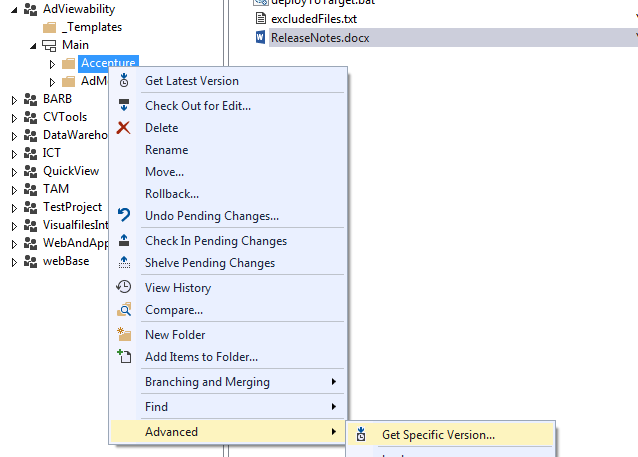
Comments to accompany the changeset are mandatory – these are for the benefit of developers, not customers, so can be technically focused.

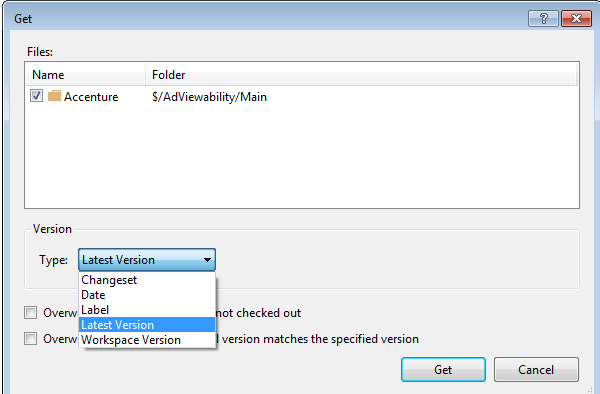
Work items can also be associated with changesets (work item id’s are obtainable from the My Work section of the Team Explorer panel, or the web interface under the Work tab). Selecting Resolve at this point will also close the work item as part of the Check In operation. More detail under the Work Items section of this document.

***Rolling Back***

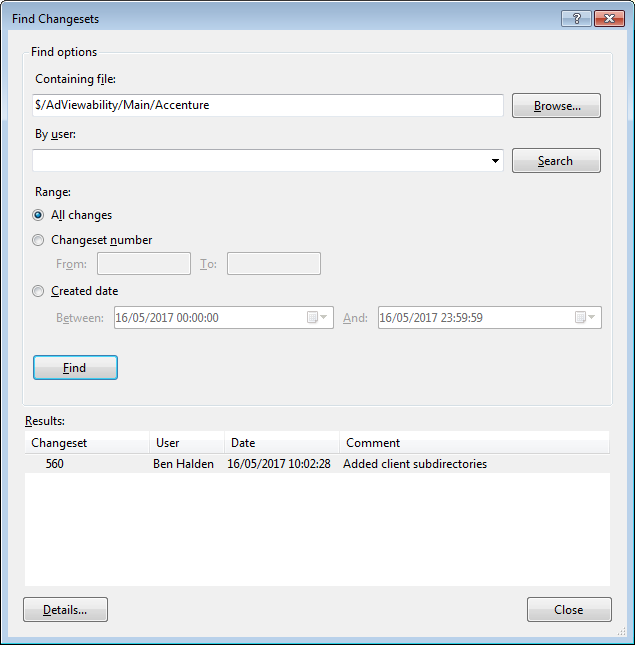
To roll files back to an earlier version, you must first identify the relevant changeset. This can be done through the web interface under the Code > Changeset tab.

Alternatively retrieve older versions of the files through Advanced > Get Specific Version:

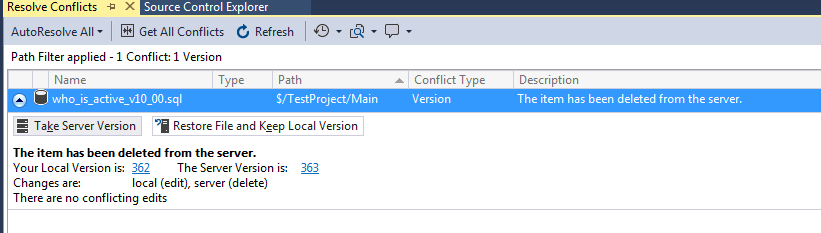




Choose changeset, and you can search the changeset history:



There is now an older version of the file(s) in your workspace. When those files are checked in, a conflict resolution will appear:

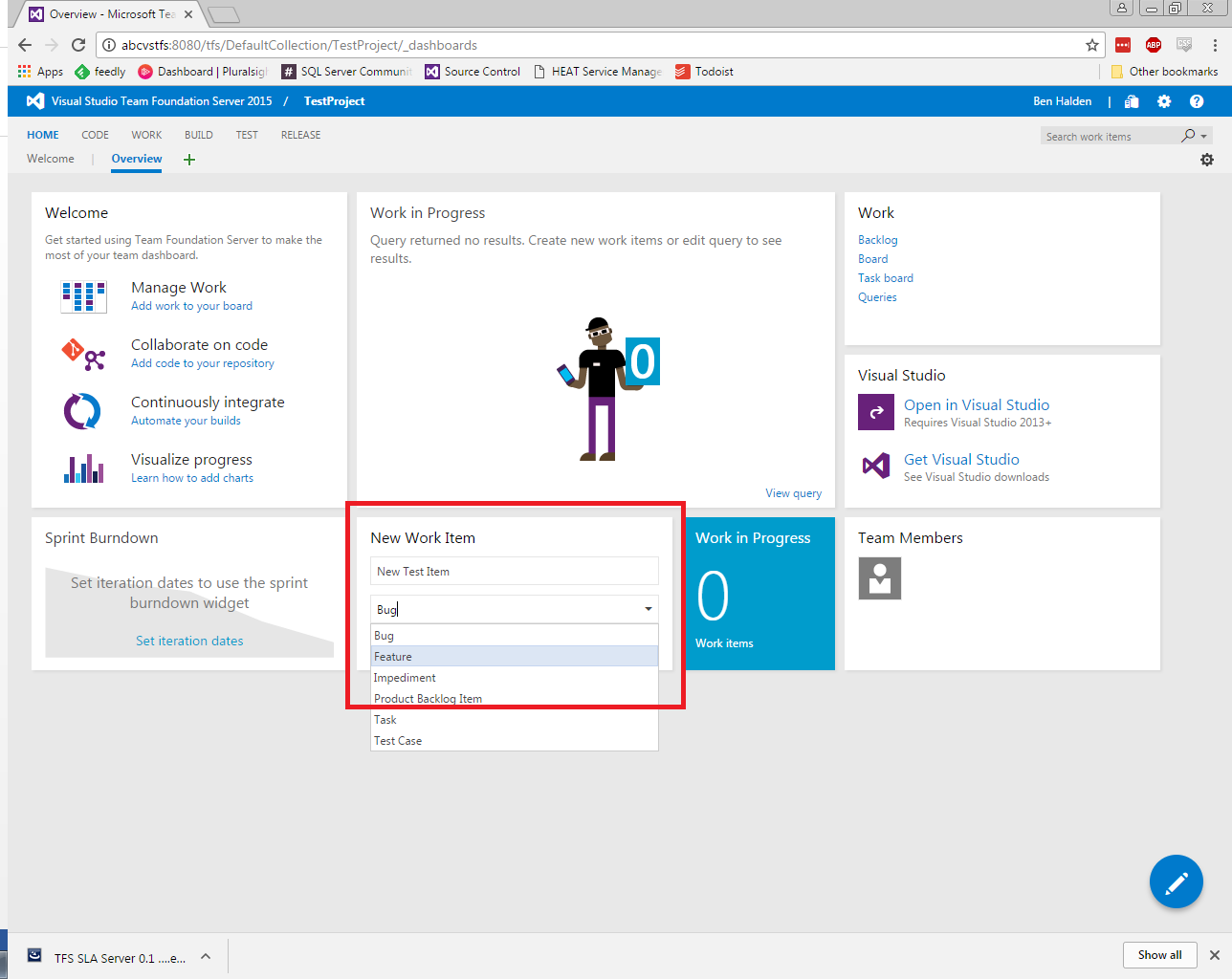


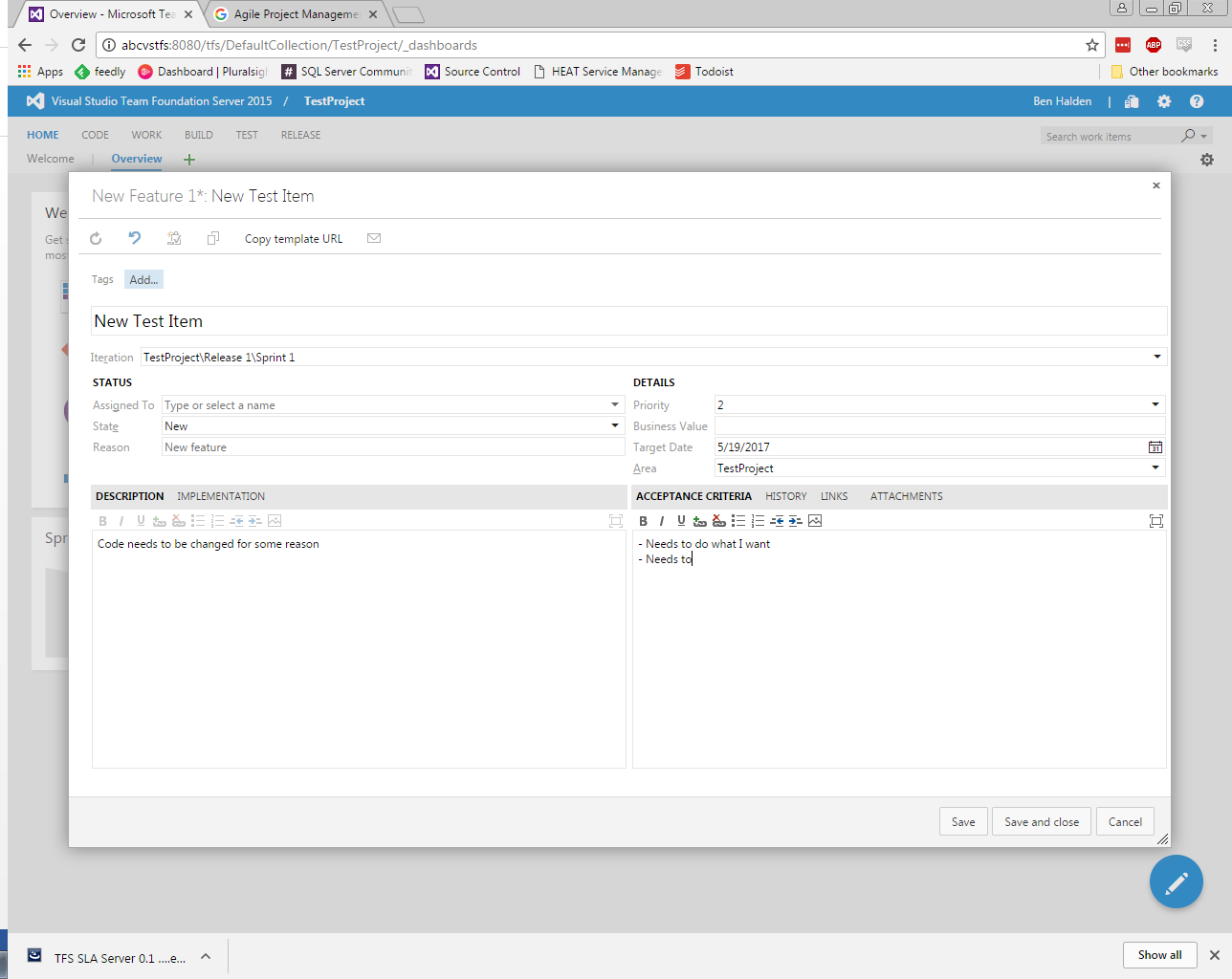
Select the appropriate option to push the workspace version back to TFS as the latest changeset. Note that no information is destroyed – the earlier versions are simply recommitted as a new changeset.

**Work Items**

*Creating*

Work items are created through the [web interface](http://abcvstfs:8080/tfs/) by navigating to the relevant team project:



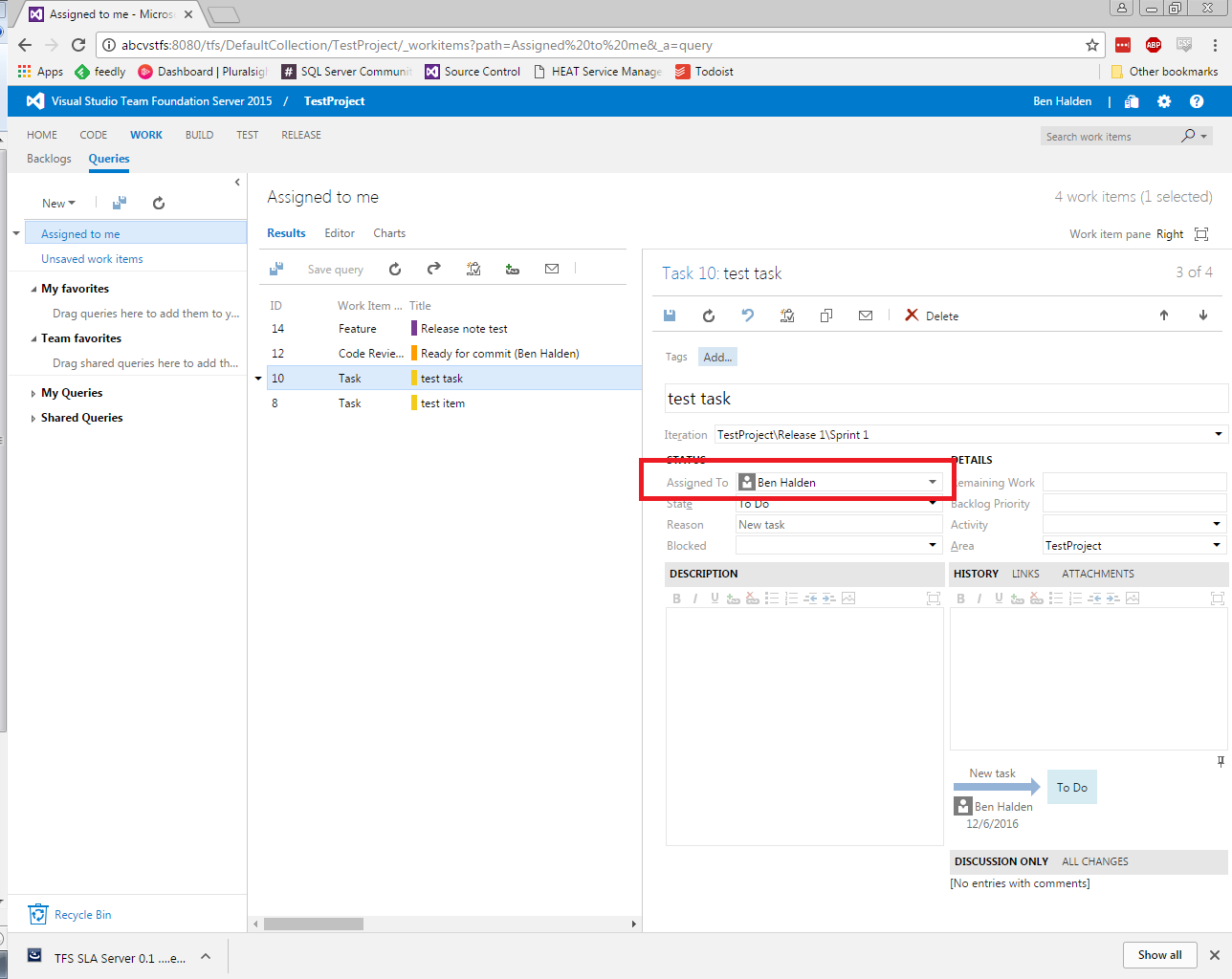


This will typically be done by customers, but creating your own work item is a good way to set out detailed requirements and track progress.

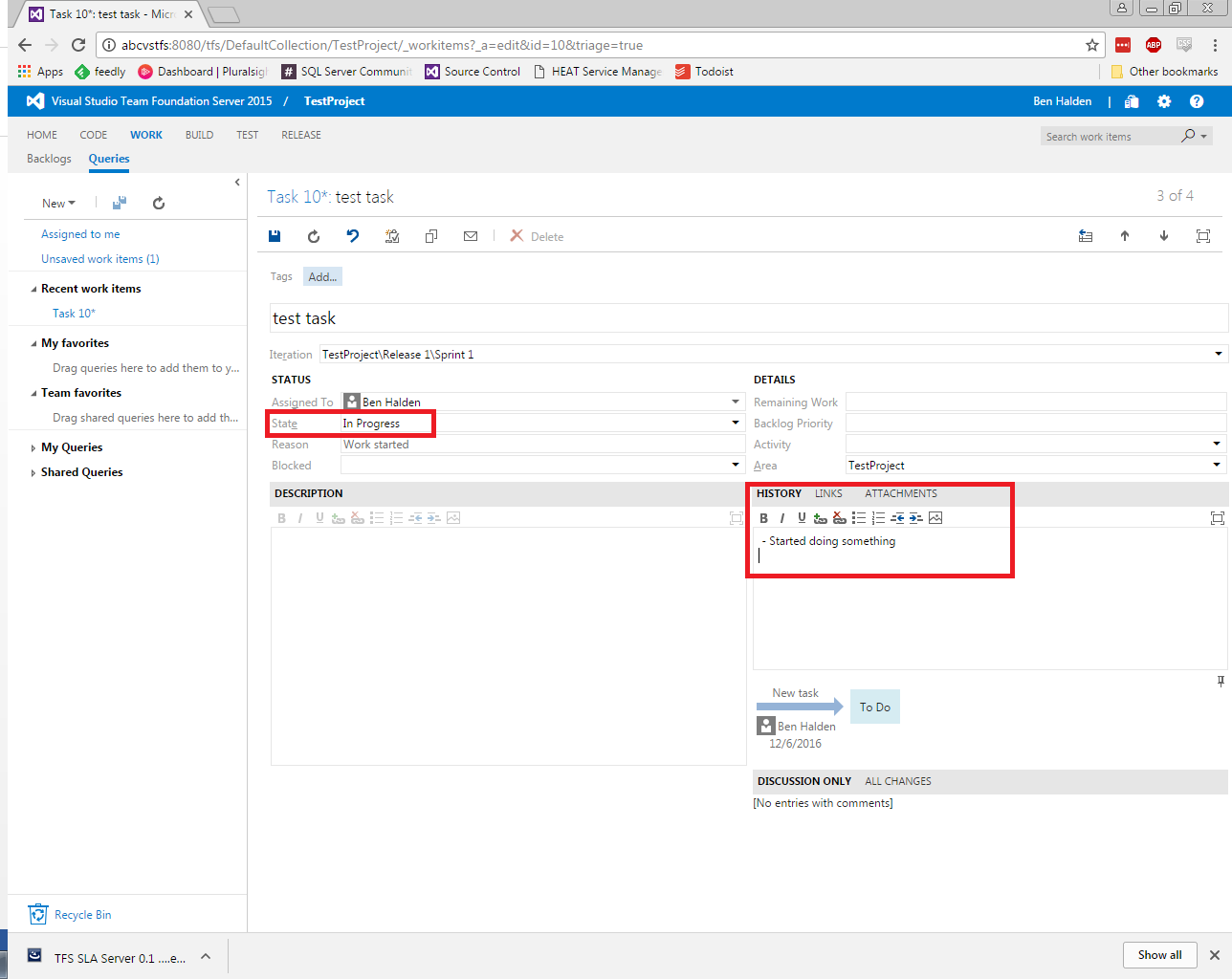
*Managing*

When a new work item is created, members of the project team will be alerted by email. A developer should take ownership of the item as soon as possible. The team will be alerted by email when the Assigned To field is changed.

Work items can be accessed through the web interface, under the “Work” tab of the relevant Team Project:



The developer can add history notes, attachments and manage the state of the item from here. The developer should update work items with notes as appropriate.



Where relevant, a work item should be associated with changesets. This is performed at check in by selecting a work item by ID. A work item may be associated with multiple changesets, so the “Resolve” option should only be selected if the changeset closes the ticket.

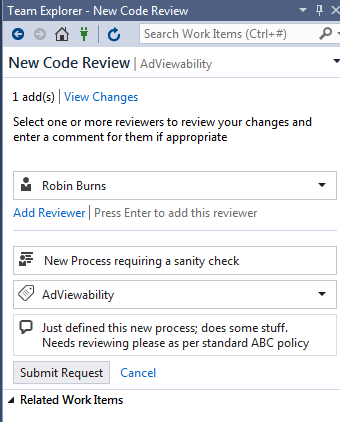
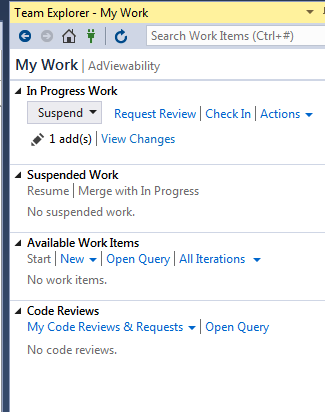
**Code Review**

To protect the codebase, a degree of quality control is required. To ensure flexibility, this is defined by policy, rather than enforced by gated check in. The guidance is as follows:

|  |  |
| --- | --- |
| **Description** | **Code Review** |
| Files added to Source Control | Not Required |
| Amendments to existing scripts and processes | Not Required |
| Newly defined / written areas of work | **Required** |

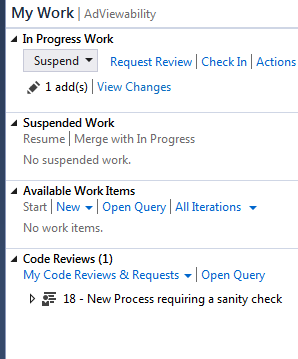
***Requesting a code review***

In order to initiate a code review, the developer must have work checked out in Visual Studio. Navigate to the My Work tab of Team Explorer in the appropriate Team Project, and request a new code review.

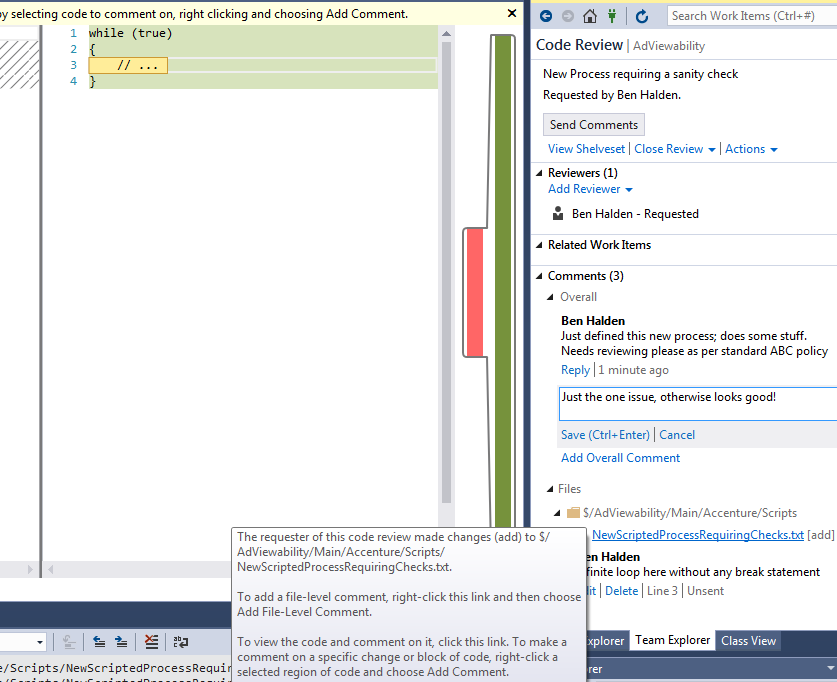


***Completing a code review***

When a code review is requested, the reviewer will be notified by email. Code reviews are viewable through the web interface, under the Work tab of the appropriate Team Project, or under the My Work tab of Team Explorer:



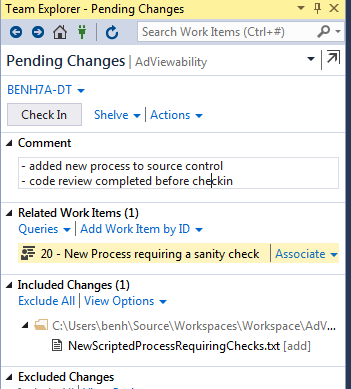
**Before beginning a code review, make sure you have no other work in progress! Commit or suspend any changes.**



The reviewer can add overall comments to the review, attach comments to individual files, or annotate lines of code within the file.

To sign off the code review, the reviewer should select “Close Review” and provide at least Overall comment. Once the code review has been closed by the reviewer, the requester may check in the pending changes:

Mandatory comments



Code review work item with comment chain automatically attached and closed when reviewer completed