Chapter 1

Source control with SVN

TechPubs uses Subversion (SVN) source control software for managing DITA files. Learn how to incorporate SVN procedures into your daily workflow.

Harmonic TechPubs has structured its deliverables to take advantage of the content reuse methods available in DITA.

- *Committing files to SVN*
- *Adding a new directory to SVN*
- *Creating snapshots after a product release*
- *Creating branches in SVN*
- *Managing SVN check-in notifications with Fisheye/Crucible*
- *Troubleshooting the "inconsistent newlines" error*
- *SVN source control: Best practices*

Committing files to SVN

Save the changes from your local working copy to the SVN repository using Tortoise SVN shortcuts in Windows Explorer.

Before you begin

Before you commit any changes, make sure that your working copy of the content folder is up to date. From Windows Explorer, select all the folders in the content folder and then right-click and select **SVN Update**.

TIP: Commit changes in logical groups or changesets. For example, if a set of files are associated with a JIRA ticket, commit those changes together.

1. Open Windows Explorer and locate your local working copy of DITA_SVN

   Example: `C:\Users\[username]\Documents\DITA_SVN\DITAUtil\content\[product]`

2. If you have created any new files, select them and go to **Tortoise SVN** > **Add**.

3. Select the files that have your changes, including any new files, right-click them, and select **SVN Commit**.
   Files that you changed have a red exclamation mark icon.

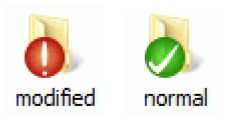
4. Write a short, clear message that indicates the changes you made and click **OK**.

   Example: Added Windows 10 client support

Example: MG-13177 added new configuration parameters for each type of authentication

Result:

After the commit has finished successfully, the icons for files and folders are updated in Windows Explorer from red exclamation marks to green check marks.



Map and topic file names

Name files in a way that will help find those files later and understand what they contain.

In general, file names follow the convention *[<type>_<product>_<description_of_content>.dita]*. The description should clearly communicate the file content. Product name is not required, however it is helpful to use in order to differentiate one topic that has a filename similar to another topic.

Other conventions:

- Use lowercase characters.
    - XML documents are case sensitive. An `href` link to `MyFile.DITA` is different from `myfile.dita`. Consistently using lowercase characters avoids confusion.
- Do not use spaces, special characters, or punctuation. To separate words use only hyphens and underscores.
- Avoid reproducing the topic title in the file name since the topic title may change.
- Avoid complex acronyms.
- Avoid unnecessarily long file names.

    NOTE: To date we have not had any issues with long file paths. Instead, the challenge has been giving files meaningful names and using a consistent naming pattern. Filenames that are clear are proving to be more valuable than filenames that are brief.

    - Some operating systems have file name character limits. For example, the maximum file path for a Windows system is 260 characters. This path length includes both the name of the folder and the name of the file itself.
    - References in DITA files might use relative paths. For example, if you reference an image that is in a folder inside of another folder, the file path can get quite long.

File Type	Standard	Example

Concept Topic	`c_`*`[product_subject]`*`.dita`
	`c_`*`[subject]`*`.dita`
		`c_electrax_system config.dita`
		`c_multiplexing.dita`

Task Topic	`t_`*`[product_action]`*`.dita`
	`t_`*`[subject_action]`*`.dita`
		`t_electraxvm_assign_ipaddress.dita`
		`t_encoder_attach_rack_rails.dita`

Reference Topic	`r_`*`[product_subject]`*`.dita`
	`r_`*`[subject]`*`.dita`
		`r_electrax_specification.dita`
		`r_electraxvm_sys_reqs.dita`

File Type	Standard	Example
Glossary Topic	`g_`*[full-name-of-term]*`.dita`	`g_transport_stream.dita.dita` `g_access_control_group.dita`

Ditamap
 	
Component map or submap used in a deliverable
	`m_`*[workflow_category]*`.ditamap`
	`m_`*[place_component]*`.ditamap`
		`m_encoding_install_sw.ditamap`
		`m_notices_copyright.ditamap`
		`m_appendix_compliance.ditamap`

Ditamap Main map of a deliverable	`d_`*[product_doctype]*`.ditamap`	`d_electrax2_installguide.ditamap` `d_electraxvm_help.ditamap`

Adding a new directory to SVN

Create a new product or other directory for the TechPubs SVN repository.

Creating a new module on the Jenkins server

Modules are subdirectories used to separate folders and files in SVN. You must create a module before you add a new directory to the TechPubs SVN repository.

Before you begin

Administrative privileges are required at different stages in this procedure.

1. Log on to *Jenkins* with your Harmonic (HLS) domain credentials.
2. Click **Create Module** > **Build with Parameters** to open the page **Project (CM) Create Module**.
3. Enter the required information for a new module.

Option	**Description**
Module_Name	Enter a name that follows this convention: `ProductName_DOC`. For example, `CableOS_DOC` or `Reuse_XVM`. Note the capitalization.
Module_Description	Describe the files that the module will contain.
Module_1st_Contact **Module_2nd_Contact** **Module Authorized Reviewers**	Enter valid Harmonic email addresses for the contacts who will receive build notifications. Code review requirements will be disabled by the build engineer after the directory is created.
SVN_Module_Group	Set to TechPubs

4. Click **Build**.

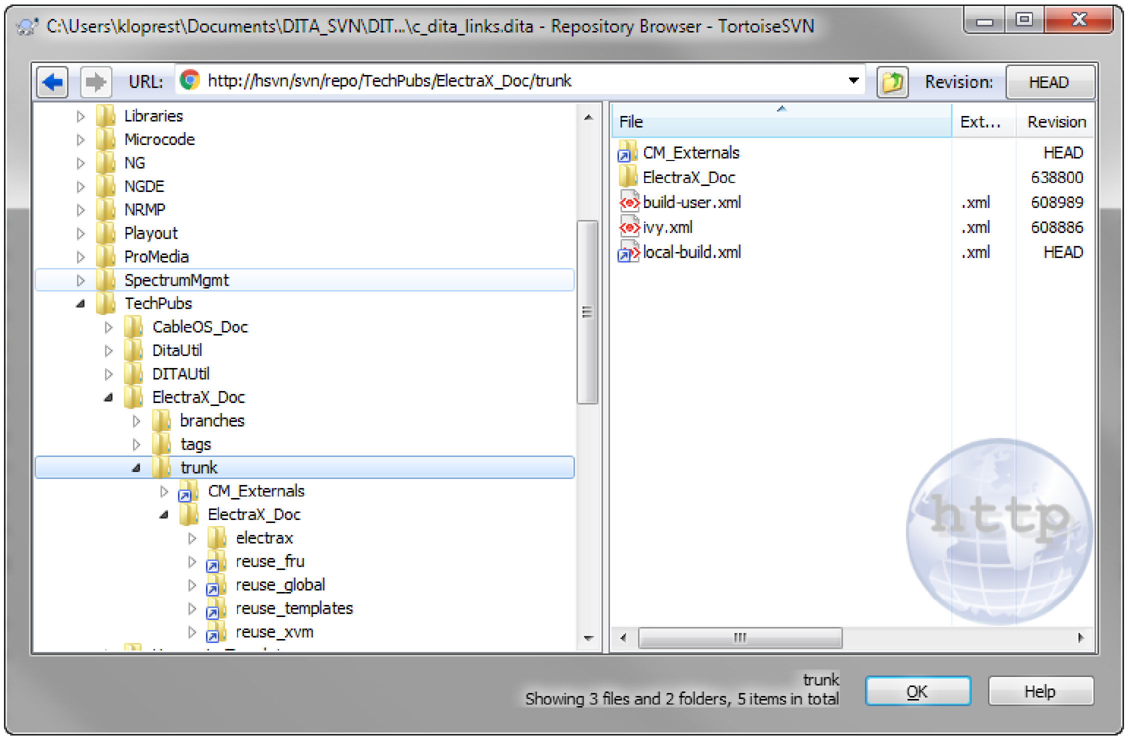
Result: The new directory is created on the Jenkins server with the name you entered and folders for /branches, /tags, and /trunk.

What to do next

Contact the build engineer ([redacted]) to let him know you created a new directory. Indicate which reuse directories the new folder uses, such as reuse_global or reuse_fru. The build engineer will link these reuse folders, known as *SVN externals*, to the product folders. After the SVN externals have been configured, create the new product folder.

DITAUtil repository structure

The TechPubs SVN repository, also known as DITAUtil, contains content collections for each product, transform scripts, templates, as well as team training and administrative materials.



C:\Users\kloprest\Documents\DITA_SVN\DIT...\c_dita_links.dita - Repository Browser - TortoiseSVN

URL: http://hsvn/svn/repo/TechPubs/ElectraX_Doc/trunk Revision: HEAD

```
Libraries
Microcode
NG
NGDE
NRMP
Playout
ProMedia
SpectrumMgmt
TechPubs
    CableOS_Doc
    DitaUtil
    DITAUtil
    ElectraX_Doc
        branches
        tags
        trunk
            CM_Externals
            ElectraX_Doc
                electrax
                reuse_fru
                reuse_global
                reuse_templates
                reuse_xvm
```

File	Ext...	Revision
CM_Externals		HEAD
ElectraX_Doc		638800
build-user.xml	.xml	608989
ivy.xml	.xml	608886
local-build.xml	.xml	HEAD

trunk
Showing 3 files and 2 folders, 5 items in total

Figure 1-1: DITAUtil repository structure from the Repository Browser

Item	Folder Name	Contents
1	`branches`	A branch is a side line of development which is created when a writer must make changes that could disrupt the main line of development (currently unused by TechPubs).

Item	Folder Name	Contents
2	`tags`	A tag is a label to highlight a notable revision in the history of the repository, such as a product release. Tags are not intended as a line of development; however, you can commit changes to a tag if necessary. A *snapshot* is a particular type of tag, which TechPubs uses to capture the state of DITA topics at a product release.
3	`trunk`	Trunk is the main line of development, and typically the level at which you check out the repository.
4	`CM_Externals`	Contains local build XML files. Writers do not make changes to these files.
5	`DITAUtil`	Contains the DITA, CSS, and for generating deliverables
6	`content`	Contains product folders for all DITA maps and topics, as well as team-related materials and topic and map templates.
7	`css`	Contains CSS files for HTML output. Writers do not make changes to these files.
8	`dita-ot`	Contains all of the DITA OpenToolkit plugins used for executing transformations in Oxygen. Writers do not make changes to these files.
9	`transforms`	Contains the transform scenarios for each type of output. Writers do not make changes to these files.

Creating a product folder for a new SVN module

After you create a new SVN module, create a folder that stores product images, maps, topics, and reuse files in SVN.

Before you begin

Creating a new module on the Jenkins server

1. Open the TortoiseSVN Repository Browser and browse to the TechPubs directory `http://hsvn/svn/repo/TechPubs`.
2. Find the new directory you created when you created the new module and browse to `/trunk/`*`[ProductName_DOC]`*.

   Result: If any SVN externals such as reuse_global or reuse_xvm were configured for the new module, those folders appear here.
3. Add a new product folder.
   a. Right-click the folder at `/trunk/`*`[ProductName_DOC]`* and select **Create Folder**.
   b. In the **New name** box, type the product folder name in lower case and click **OK**.
   c. Enter a commit message and click **OK**.
   d. In the product folder, create the required subfolders that follow the TechPubs *SVN content directory structure on page 9*.

What to do next

Contact the build engineer ([redacted]) to let him know you created a new product folder. Ask to add it to `content_checkout.bat`, the script used to check out working copies of TechPubs DITA source files.

Checking out a new product folder

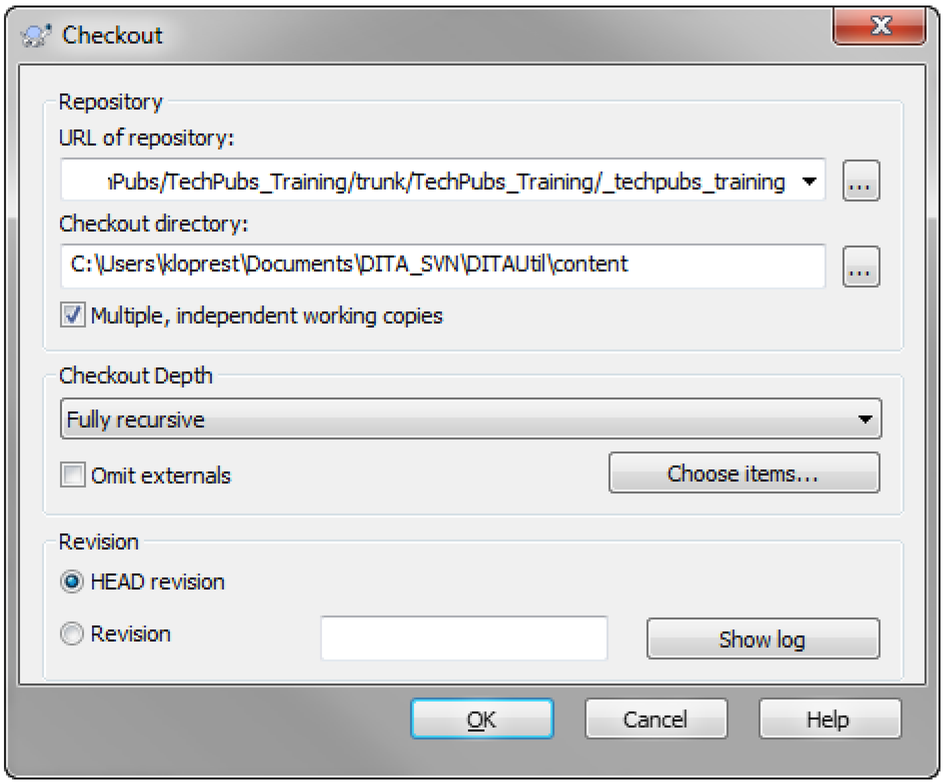
Add a new product folder to your existing working copy of DITA_SVN.

Before you begin

Creating a new module on the Jenkins server
Creating a product folder for a new SVN module

1. Check in any uncommitted changes in your working copy of DITA_SVN.
2. From Windows Explorer, go to `C:\Users\`*`[user name]`*`\Documents\DITA_SVN\DITAUtil\`
3. Right click the file `content_checkout.bat` and select **SVN Update**.
4. Open the file `content_list.txt` and find the new product folder name. If it is not there, add the to the list. Save your changes.
5. Double-click `content_checkout.bat`.

   Result: The **Checkout** window opens.


Checkout
Repository
URL of repository: ıPubs/TechPubs_Training/trunk/TechPubs_Training/_techpubs_training
Checkout directory: C:\Users\klopresti\Documents\DITA_SVN\DITAUtil\content
[✓] Multiple, independent working copies
Checkout Depth
Fully recursive
[ ] Omit externals
Revision
(●) HEAD revision
( ) Revision

6. Confirm the following default values.

Option	Description
Checkout directory	`C:\Users\[user name]\Documents\DITA_SVN\DITAUtil\content`
Checkout depth	**Fully recursive**
Revision	**HEAD revision**

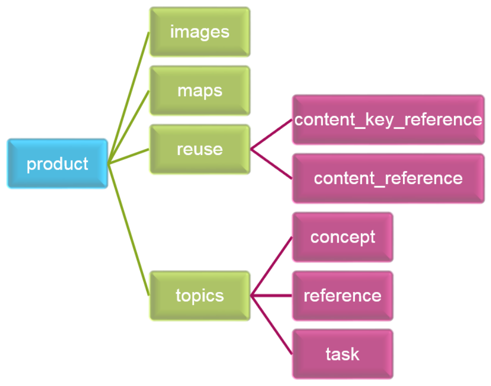
7. Select the check box **Multiple, independent working copies** and click **OK**. The Target folder warning opens.
8. Confirm the checkout to the non empty target folder.

   Result: TortoiseSVN begins to download copies of DITA files to your working directory. Any checked out directories will undergo an "update" and new directories will be added. This may take a few minutes.
9. When the process completes, open the `C:\Users\[user name]\Documents\DITA_SVN\DITAUtil\content` and verify that you have downloaded the files that you need. If you are missing any directories, make sure they are included in the `content_list.txt` file and run `content_checkout.bat` again.

SVN content directory structure

TechPubs stores all DITA maps and topics in the `content` directory. This directory has been structured to track multiple types of content in a consistent way and to take advantage of the content reuse methods available in DITA.

A product folder, also known as a *content collection*, contains all of the images, topics, and DITA maps that comprise the product's various deliverables. Each product also has a designated `reuse` folder, which contains master content reference and content key reference files.



Creating snapshots after a product release

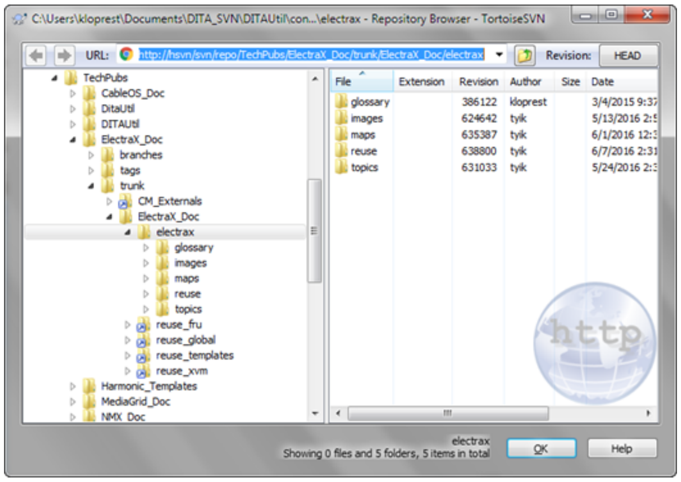
SVN snapshots capture files at a certain point in time. After a product release or publication of a deliverable, create a snapshot to record the milestone. These files are separate from the files in SVN trunk and are used to recreate or update previously released output.

Before you begin

Check in any uncommitted changes in your content directory.

1. In Windows Explorer, from your local working copy of DITA_SVN, right-click the folder you want to capture and select **Tortoise SVN** > **Repo-browser**.

   Result: The **Repository Browser** window opens.

   C:\Users\kloprest\Documents\DITA_SVN\DITAUtil\con...\electrax - Repository Browser - TortoiseSVN

   URL: http://hsvn/svn/repo/TechPubs/ElectraX_Doc/trunk/ElectraX_Doc/electrax   Revision: HEAD

   ```
   TechPubs
     CableOS_Doc
     DitaUtil
     DITAUtil
     ElectraX_Doc
       branches
       tags
       trunk
         CM_Externals
         ElectraX_Doc
           electrax
             glossary
             images
             maps
             reuse
             topics
           reuse_fru
           reuse_global
           reuse_templates
           reuse_xvm
     Harmonic_Templates
     MediaGrid_Doc
     NMX_Doc
   ```

File	Extension	Revision	Author	Size	Date
glossary		386122	kloprest		3/4/2015 9:37
images		624642	tyik		5/13/2016 2:
maps		635387	tyik		6/1/2016 12:
reuse		638800	tyik		6/7/2016 2:31
topics		631033	tyik		5/24/2016 2:

   electrax
   Showing 0 files and 5 folders, 5 items in total

2. Right-click the parent directory of the folder you want to capture and click **Copy to**.

   Example: If you are creating a snapshot for the folder `electrax`, copy the source files from the parent directory `ElectraX_Doc`.

3. In the **New name** field, type the destination URL for the snapshot.

   Example: `http://hsvn/svn/repo/TechPubs/ElectraX_Doc/tags/_snapshots/1.3.0.0`
   If a product has multiple devices and nonlinear releases, add a device directory to the path.

   Example: `http://hsvn/svn/repo/TechPubs/ElectraX_Doc/tags/_snapshots/electra_x2/1.3.0.0`

4. Enter a **Log message** to describe the product release or other document milestone.

5. Verify that the snapshot you created contains the content you want to capture.
   a. From Tortoise SVN, click **Repo-browser** and navigate to `TechPubs/`*`[product_Doc]`*`/tags/_snapshots`.
   b. Drill-down to the destination folder of your snapshot.
   c. Verify that all the maps, topics, images and other files have been copied to the new folder.

Creating branches in SVN

Create an SVN branch to isolate changes onto a line of development separate from trunk.

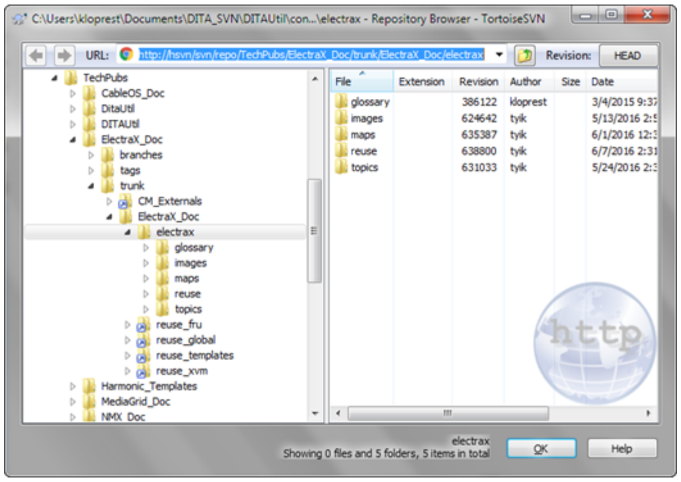
Before you begin

Check in any uncommitted changes in your content directory.

Branches are used to test features or make changes without disturbing the main line of development. As soon as the new feature is stable enough then the development branch is merged back into the main branch (trunk).

1. In Windows Explorer, from your local working copy of DITA_SVN, right-click the folder you want to branch and select **Tortoise SVN** > **Repo-browser**.

   Result: The **Repository Browser** window opens.



C:\Users\kloprest\Documents\DITA_SVN\DITAUtil\con...\electrax - Repository Browser - TortoiseSVN

URL: http://hsvn/svn/repo/TechPubs/ElectraX_Doc/trunk/ElectraX_Doc/electrax Revision: HEAD

File	Extension	Revision	Author	Size	Date
glossary		386122	kloprest		3/4/2015 9:37
images		624642	tyik		5/13/2016 2:5
maps		635387	tyik		6/1/2016 12:3
reuse		638800	tyik		6/7/2016 2:31
topics		631033	tyik		5/24/2016 2:3

electrax
Showing 0 files and 5 folders, 5 items in total

2. Right-click the trunk directory of the folder you want to branch and click **Copy to**.
3. In the **New name** field, type the destination URL for the branch.

   Example: `http://hsvn/svn/repo/TechPubs/NMX_Doc/branches/7.5.1.0`.
   If a product has multiple devices and nonlinear releases, add a device directory to the path.`http://hsvn/svn/repo/TechPubs/ElectraX_Doc/branches/electra_x2/1.3.0.0`
4. Enter a **Log message** to describe the branch.

   Result: A new branch folder is created.
5. Verify that the branch you created contains all the directories included in the original trunk directory.
   a. From Tortoise SVN, click **Repo-browser** and navigate to `TechPubs/`*`[product_Doc]`*`/branches`.
   b. Drill-down to the destination folder of your branch.
   c. Verify that all the maps, topics, images and other files have been copied to the new folder.

Managing SVN check-in notifications with Fisheye/Crucible

Receive an e-mail notification for file check-ins to a DITA_SVN directory.

1. Log on to FishEye & Crucible and open the TechPubs repository at [redacted]/ *hsvn.repo.TechPubs*
2. Navigate to a product or other TechPubs directory such as TechPubs_Standards or reuse_global.
3. Choose **Tools** > **Watch**.

   Result: The page reloads and a watch is set up for the activity stream.

Troubleshooting the "inconsistent newlines" error

Sometimes, when attempting to add an .svg image to SVN, the operation will fail with an error indicating "inconsistent newlines." You can use an online tool to remove line breaks from the .svg file.

This seems to happen more frequently when an older .eps or .ai file is saved as .svg.

1. Open a browser and navigate to the site *textfixer.com* and find the online tool **Remove Line Breaks**.
2. Open the .svg file with a text editor such as Notepad.
3. Select all text (**Ctrl**+**A**) and copy (**Ctrl**+**C**).
4. Paste the contents of the file into *textfixer.com* and click **Remove Line Breaks**.
5. Copy all of the output, then paste over the contents of the source file in your editor.
6. Save the .svg file.

   Result: You may now add and commit the image file to SVN.

SVN source control: Best practices

Review best practices for working in the SVN repository and collaborating with team members.

Which SVN method to use?

- As a standard, use the Copy-Modify-Merge model when editing .dita files.
- Use the Branch and Merge model only when necessary.
- When editing graphics, you may use the Lock-Modify-Unlock model to prevent other writers from making changes to the same graphic file at the same time. Remember to unlock the file when finished.

SVN updates

- Update your local working copy each morning, and again before committing changed files.

Collaboration

- Email the team when making changes to a file in **reuse_global**.

- If you are working on files that are shared with other writers, advise the other writers of the changes you plan to make. Be sure to email those writers when:
    - changes you make could break a link (for example, renaming a shared file,deleting a shared file, or changing an element ID that is reused through a content reference).
    - changes you make might prevent a variable from resolving.
    - you reorganize a shared component map.
- If an SVN conflict occurs, notify the other writer(s) involved and work to resolve the conflict.

SVN commits

- Write a *short*, clear message indicating the changes you made to the files being committed. For example: "Updated the topic titles to follow sentence-style capitalization."
- Commit changes at or below the product folder level.
- Commit changes in logical groups or changesets. For example, if a set of files are associated with a JIRA ticket, commit those changes together. Or, commit changes for the *Electra X2 Installation Guide* separately from changes for the *Electra XVM Installation Guide*.
- Commit at the file level if you want to tailor your commit message to a specific file.
- Avoid last-minute check-ins of shared files before vacation. Leave enough time to resolve potential conflicts with other writers before you leave.

Publishing

- After publishing a deliverable, create a snapshot of the folders referenced by the deliverable map.

Check-in notifications

- Use Fisheye/Crucible to manage notifications of check-ins to the DITAUtil repository.