# SCRIVENER FILE FORMAT SPECIFICATION

# VERSION 1.6

**13 October 2014**

# Contents

# REVISION HISTORY

| Date | Version | Description | Author |
|---|---|---|---|
| 20 April 2010 | 1.0 | First Draft | Keith Blount |
| 28 April 2010 | 1.01 | **Added:** Image symbolic links | Keith Blount |
| 19 May 2010 | 1.02 | **Added:** Note on RTF line breaks. | Keith Blount |
| 27 May 2010 | 1.03 | **Added:** AllowNegatives attribute to the SessionTarget SCRIVX element. | |
| 26 July 2010 | 1.04 | **Updated:** Added note clarifying that the lowest ID for project notes should be 10.<br><br>**Updated:** File formats for snapshots and project notes have been changed. Both snapshots and project notes are now stored as folders containing multiple RTF files and an index.xml file, to avoid problems parsing large XML files containing RTF with the Cocoa parser. (These problems do not affect the RTF stored in .links files, which remains as-is.) | Keith Blount |
| 8 September 2010 | 1.05 | **Added:** NotesTextSelection to the BinderItem Element. | |
| 17 October 2010 | 1.06 | **Changed:** CapsDelimiter element of the script settings XML format renamed to StyleDelimiter | Keith Blount |
| 21 January 2011 | 1.07 | **Added:** Creator attribute to main ScrivenerProject element.<br><br>**Added:** PlaybackPosition to the MediaSettings element of BinderItem.<br><br>**Added:** TextColor to the CharacterFormat element of the script settings XML format. | Keith Blount |
| 16 March 2011 | 1.08 | **Added:** Options to the **ProjectTargets** element for enhanced session targets. | Keith Blount |
| 15 April 2011 | 1.09 | ~~**Added:** ExcludeWeekends attribute to the SessionTarget element.~~ (Removed 02/05/11)<br>**Added:** IsAlias to BinderItem media settings. | Keith Blount |

| | | | |
|---|---|---|---|
| 2 May 2011 | 1.10 | **Removed:** ExcludeWeekends attribute from SessionTarget element.<br><br>**Added:** WritingDays attribute to SessionTarget element. | Keith Blount |
| 19 May 2011 | 1.11 | **Updated:** The "Type" attribute of the "SearchSettings" element in collection elements can now contain multiple values, to facilitate searches across more than one part of the document. | Keith Blount |
| 9 June 2011 | 1.12 | **Added:** The user.lock file now has an "app" value for reporting to the user which other program is using the file. | Keith Blount |
| 1 July 2011 | 1.2 | **Added:** "Scope" attribute to "Completion" element in the SCRIVX file.<br><br>**Added:** "FootnoteNumber" attribute to the "Comment" element in the .links XML file. | Keith Blount |
| 9 July 2011 | 1.21 | **Added:** "ProjectProperties" element to SCRIVX file. | Keith Blount |
| 18 July 2011 | 1.22 | **Added:** "UserData" element to BinderItem in the SCRIVX file. | Keith Blount |
| 22 August 2011 | 1.23 | **Updated:** The "ResetDaily" attribute of the "SesionTarget" element has been replaced with the "ResetType" element. | Keith Blount |
| 19 September 2011 | 1.24 | **Added:** "IsSoundFile" element to MediaSettings sub-element of BinderItem. | Keith Blount |
| 4 October 2011 | 1.25 | **Added:** "DefaultChildTemplateID" element to BinderItem. | Keith Blount |
| 3 November 2011 | 1.26 | **Added:** "CanWriteOnDeadlineDate" attribute to SessionTarget element. | Keith Blount |
| 28 March 2012 | 1.27 | **Added:** "GroupMode" element to "BinderItem" and "Collection" elements. | Keith Blount |
| 1 June 2012 | 1.28 | **Added:** "BookmarkData" element to "FullScreenBackdrop" element in the SCRIVX file. | Keith Blount |
| 26 June 2012 | 1.29 | **Added**: Information on "Preserve Formatting" blocks to "RTF Inline Mark-Up". | Keith Blount |
| 9 January 2013 | 1.3 | **Changed:** The way linked comments and Scrivener Links are saved and read has been altered. See p.102. | Keith Blount |

| | | | |
|---|---|---|---|
| 22 January 2013 | 1.4 | **Updated:** The way Scrivener recognises where to read link ranges from has been altered to allow for better backwards compatibility with Windows versions. | Keith Blount |
| 25 June 2013 | 1.5 | **Added:** “AllowOver” and “Buffer” attributes to the “Target” element of the “BinderItem” element.<br><br>**Added:** “ShowOverrun” and “Buffer” attributes to the “DraftTarget” element of the SCRIVX file. | Keith Blount |
| 19 September 2014 | 1.6 | **Updated File Format:**<br>- SCRIVX “Version” changed to 1.5.<br>- version.txt version changed to 18.<br>- “ScrivenerProject” element of SCRIVX file has new attributes: Device, Modified, ModID, SyncID and Checksums.<br>- “BinderItem” element of SCRIVX file has new attribute: UUID<br>- “TemplateID” element of SCRIVX has been changed to “TemplateUUID”<br>- .links files are no longer used to save Scrivener links and comments. Instead, Scrivener links are saved using in-RTF custom URLs of the format “scrivlnk://“ and comments are saved in .comments XML files.<br>- Removed “Links” subelement of “Note” element in the Project Notes XML file. | |

# *1. Introduction*

This document describes the file format for Scrivener 2.0. It assumes you are familiar with XML conventions and the Scrivener software.

Scrivener for the Mac's file format is package-based, which means that it uses a folder of files that appears to the user as a single file. The file package the user sees is a SCRIV (.scriv) file. On other platforms, this package will appear as a regular folder, so users on other platforms will be able to see the files that comprise a SCRIV file (and Mac users can see them by ctrl-clicking on a .scriv file in the OS X Finder and selecting "Show Package Contents"). Because a SCRIV file will appear as a regular directory beyond the Mac platform, the .scriv extension is not necessary on other platforms.

The SCRIV package contains several folders of files and a SCRIVX file. The SCRIVX file is an XML file that adheres to the XML standard, and this is the file that Scrivener uses to store the main project structure data. When a Mac user opens a SCRIV file, Scrivener looks inside the SCRIV package (folder) for the SCRIVX file and opens that. On platforms that don't support file packages - such as Windows - Scrivener should open the SCRIVX file directly (Scrivener for the Mac can also open SCRIVX files directly if necessary). Note, however, that a SCRIVX file is always supported by several other files and folders as specified in this document, and does not contain *all* project data (for example, all text content is stored in discrete RTF files).

The SCRIVX file is designed to be flexible and so allows third-party software to read it and add its own data to it (according to the guidelines specified at the end of this document) then save it again. Scrivener will preserve such data when it subsequently opens and saves the file.

# *2. The Opening Behaviour of Scrivener Projects*

Scrivener can open two file types:

.scriv
.scrivx

### .scriv

.scriv files are the Mac's package file format. A .scriv file is really a folder full of files, but Mac OS X's package support means that, to the end-user, .scriv files will appear and act the same as single files. However, a user can ctrl-click on a .scriv file in the Finder and select "Show Package Contents" to reveal the files contained inside the .scriv folder.

When a Mac user opens a Scrivener project, he or she will generally open the .scriv file, either by double-clicking on it in the OS X Finder or by opening it from within the program.

Unfortunately Windows does not support package files, so a .scriv file will appear as a regular folder on Windows despite its file extension.

### .scrivx

The .scrivx file is an XML file contained in the root of the .scriv folder which stores the main binder structure and other key project information. (I suggest that the Windows version allow double-clicking on this file in Windows Explorer to open the project.)

If a Windows user renames the .scriv folder so that the .scriv extension is no longer present, should the file then be moved to a Mac it will no longer appear as a file package (because packages rely on file extensions), and will appear on OS X, too, as a folder. In this circumstance, Scrivener for the Mac also allows the opening of .scrivx files. If a user on OS X goes to open a .scrivx file, Scrivener for the Mac will inform him or her that it will automatically add the .scriv extension to the enclosing folder, which is necessary for the Mac version to open the file.

In this way it should be possible - by supporting both the .scriv wrapper (predominantly Mac) and the .scrivx XML file (predominantly Windows) - for projects to be moved smoothly between platforms with a minimum of fuss for the user.

# *3. The Structure of a SCRIV File Package*

This chapter describes the structure of a .scriv file, specifying all of the files and folders that appear inside a .scriv project folder.

A SCRIV file has a particular structure that must be adhered to in order for a Scrivener project to be valid and thus successfully read. The Scrivener file format essentially comprises a folder of files and subdirectories, some of which must be present and others of which are only created as needed. On the Mac, the root folder containing all of these files and subdirectories appears to the user as a single SCRIV (.scriv) file, because Mac OS X has a "file package" feature whereby a folder may be treated by the system, and appear to the user, as a single file. On other platforms, the .scriv extension may or may not be present, but whether it is present or not, this root package will appear to users on other platforms as a regular directory.

This root directory - which has the file extension .scriv and appears as a single file on the Mac but which appears as a regular folder on other platforms - stores all information pertaining to a single Scrivener project and is referred to throughout this document interchangeably as a SCRIV directory, file, folder or package.

The internal structure of a SCRIV directory is as follows:

| **File** | **Type** | **Required** |
|---|---|---|
| [Project Name].scrivx | XML file | Yes. |
| Files | Directory | Yes. |
| - user.lock | Plain text file | When open. |
| - version.txt | Plain text file | Yes. |
| - binder.backup | Zip file | No. |
| - binder.autosave | Zip file | No. |
| - search.indexes | XML file | Yes. |
| - ProjectNotes | Directory | No. |
| - index.xml | XML file | No. |
| - Note-[ID].rtf | RTF file | No. |
| - Docs | Directory | Yes. |
| - docs.checksum | Plain text file | Yes. |
| - [ID].[ext] | File | No. |
| - [ID]_notes.rtf | RTF file | No. |
| - [ID]_synopsis.txt | Plain text file | No. |
| - [ID]_icImage.[ext] | Image file | No. |
| - [ID].comments | XML file | No. |
| Snapshots | Directory | No. |
| - [ID].snapshots | Directory | No. |
| - index.xml | XML file | No. |
| - [DATE].rtf | RTF file | No. |
| Settings | Directory | Yes. |

| | | |
|---|---|---|
| - ui.plist | Property list | No. |
| - projectpreferences.xml | XML file | No. |
| - scriptformat.xml | XML file | No. |
| - templateinfo.xml | XML file | No. |
| - compile.plist | Property list | No. |
| - pagesetup.plist | Property list | No. |
| Icons | Directory | No. |
| QuickLook | Directory | No. |

A fuller description follows:

| File | Type | Req. | Description |
|---|---|---|---|
| [Project Name].scrivx | XML file | Yes. | The SCRIVX file is the main XML file containing the binder structure, label and status settings etc. This is the file non-Mac users will double-click to open a project. (Mac users can also open a project from this file if the .scriv extension is missing from the folder and it therefore looks like a regular directory rather than a single file (package) - for instance, if the user is opening a file originally created on Windows. The file name can be anything set by the user (on the Mac, it will be the same as the SCRIV file name, although Scrivener for the Mac doesn't rely on this always being the case). |
| Files | Directory | Yes. | The folder that contains all of the main document files. |
| - user.lock | Plain text file | When open. | A lock file that tells Scrivener whether this project is already in use by another user or if Scrivener crashed the last time it was open. |

| | | | |
|---|---|---|---|
| - version.txt | Plain text file | Yes. | A plain text file encoded in UTF8 that contains a number representing the version of Scrivener that the project was saved in. Note that this version number is the *project* version currently in use by Scrivener and does not correspond to the application version number that the user sees in Scrivener's About panel. For example, Scrivener 2.0's project number is 16, so the version.txt file of projects saved in the first version of Scrivener 2.0 will contain the number "16". |
| - binder.backup | Zip file | No. | The binder.backup file is a zipped-up copy of the main SCRIVX file that is created when the project is successfully opened. It is made as a precaution, in case the main SCRIVX file becomes corrupted. If the SCRIVX file *does* become corrupted so that it cannot be opened, Scrivener will first look for and try to open the binder.autosave backup file and then, if that fails too, for the binder.backup file, which should be guaranteed to work given that it was made only after the project was successfully opened. |
| - binder.autosave | Zip file | No. | The binder.autosave file is a zipped-up copy of the main SCRIVX file that is created or updated whenever the project is saved. It is made as a precaution, in case the main SCRIVX file becomes corrupted. If the SCRIVX file *does* become corrupted so that it cannot be opened, Scrivener will first look for and try to open the binder.autosave backup file and then, if that fails too, for the binder.backup file. The binder.autosave file is a copy of the last successfully saved version of the SCRIVX file, and is created immediately before writing the new version to disk. |

| | | | |
|---|---|---|---|
| - search.indexes | XML file | Yes. | An XML file used to store all text contained in the project in plain text formatting purely for search purposes. This file is loaded into memory whenever a project is opened and saved when the project is closed again. If, upon opening a project, the docs.checksum file indicates that changes have been made to documents in the project since the project was last successfully closed, the search indexes will be updated with the text of any changed documents before the project is opened. |
| - ProjectNotes | Directory | No. | A subfolder of \Files containing the project notes. The project notes information is encoded in an XML file entitled index.xml and the text of each note is saved as an RTF file using the file name "Note-[ID].rtf", where "[ID]" represents the project note ID as defined in the index.xml file. |
| - Docs | Directory | Yes. | A subfolder of \Files containing all of the RTF, PDF files etc - all of the actual document contents - as numbered files, with the numbers representing internal IDs (which are defined in the SCRIVX file). |
| - docs.checksum | Plain text file | Yes. | A plain text file containing checksums for all RTF and TXT files inside the \Files\Docs folder as SHA1 values. This is used by Scrivener for the Mac to check its search.indexes file is up to date whenever it opens a project (rebuilding the search indexes if not). |

| | | | |
|---|---|---|---|
| - [ID].[ext] | File | No. | The main content for each document in the project is saved inside the \Files\Docs folder using an internal ID number as the filename (as defined in the SCRIVX file). The file extension for text files is always RTF. Thus there may be files with names such as 4.rtf, 5.rtf, 6.pdf, 7.mov, 8.rtf, 9.jpg and so on. Note that text documents with no content do not have a file associated with them until content is created. |
| - [ID]_notes.rtf | RTF file | No. | Any document in a project can have notes associated with it. All notes files are saved as RTF files using the internal ID number of the document (as defined in the SCRIVX file) and the suffix "_notes" as the file name. E.g. 5_notes.rtf, 6_notes.rtf etc. Note that only documents with content in their notes will have associated notes files in the \Files\Docs folder. |
| - [ID]_synopsis.txt | Plain text file | No. | Any document in a project can have a synopsis associated with it. All synopses are saved as UTF8 plain text (TXT) files using the internal ID number of the document (as defined in the SCRIVX file) and the suffix "_synopsis" as the file name. E.g. 5_synopsis.txt, 6_synopsis.txt etc. Note that documents with empty synopses will not have synopsis files in the \Files\Docs folder. |

| | | | |
|---|---|---|---|
| - [ID]_icImage.[ext] | Image file | No. | In Scrivener 2.0, any document in a project can have an image associated with it, which can be set to appear in the inspector (where the synopsis usually appears) or on the corkboard to represent the document (instead of a synopsis - although files can have both a synopsis and an image, and the user can choose which is displayed). These images are referred to as "index card images", and are saved as image files using the internal ID number of the document (as defined in the SCRIVX file) and the suffix "_icImage" (for "index card image") as the file name. E.g. 5_icImage.jpg, 17_icImage.png (any supported image file type may be used). |
| - [ID].comments | XML file | No. | Any text document in a project may have comments and footnotes associated with it that appear as links in the text to boxes in the inspector which contain the actual notes (2.0 only). Linked comments and footnotes are stored in XML .comments files - there may therefore be a .comments file associated with any text document in the project. These XML files are saved using the internal ID number of the document with which they are associated (as defined in the SCRIVX file) as the file name and the file extension ".comments". E.g. 5.comments, 6.comments etc. |
| Snapshots | Directory | No. | A folder containing all text document snapshots. |

| | | | |
|---|---|---|---|
| - [ID].snapshots | Directory | No. | The user can take snapshots of any *text* document in a project. All snapshots for a single text document are stored inside a single .snapshots directory in the \Snapshots folder. The document .snapshots folder uses the internal ID number of the document (as defined in the SCRIVX file) as the file name and the file extension ".snapshots". Each .snapshots directory contains an index.xml XML file encoding information about each snapshot, and a separate RTF file for the text of each snapshot using the date of the snapshot as the file name. |
| Settings | Directory | Yes. | A folder containing various settings files. |
| - ui.plist | Property list | No. | Mac-only. This file is a Mac property list XML file that stores the interface state (window size, split view positions, whether the binder is open or closed etc) using the Apple .plist format. Other platforms should store these settings in the \Settings directory using an equivalent file format. |
| - projectpreferences.xml | XML file | No. | An XML file storing Scrivener 2.0's project preferences. (Certain preferences - such as text formatting - can be saved on a per-project basis in Scrivener 2.0 for the Mac.) |
| - scriptformat.xml | XML file | No. | An XML file encoding the current scriptwriting formatting for the project. |
| - templateinfo.xml | XML file | No. | An XML file storing basic information about the template from which the current project was created (if any). This is solely for use by the "Save As Template" panel, so that its fields can be populated by default with the relevant settings. |

| | | | |
|---|---|---|---|
| - compile.plist | Property list | No. | Mac-only. This file is a Mac property list XML file that stores the current Compile settings using the Apple .plist format. There may be some differences between the Compile sheet on different platforms, so other platforms should store these settings in the \Settings directory using an equivalent file format. |
| - pagesetup.plist | Property list | No. | Mac-only. This file is a Mac property list XML file that stores the current Page Setup settings using the Apple .plist format. For the Mac version, this file saves the current margin settings, which elements of documents should be printed, index card printing orientation and so forth. Given that print options may differ between platforms, other platforms should store these settings in the \Settings directory using an equivalent file format. |
| Icons | Directory | No. | A folder used by Scrivener 2.0 to store custom binder icons. If it is present, it should contain image files (each with a size of 16x16 pixels) that can be used in the current project for folder and file icons in the binder (Scrivener 2.0 supports custom document icons). |
| QuickLook | Directory | No. | A folder containing files used on the Mac for QuickLook previews. Other platforms should ignore this folder and the files therein. |

## Note on Scrivener XML Files

Note that at this time none of the Scrivener XML formats have associated Document Type Definitions (DTDs), although this may change in the future.

In many situations in the various XML files used by Scrivener, an element or attribute may be required, or there may be restrictions on the number of elements or the values an attribute can take (all of which are specified in this document). In rare cases this can result in an invalid file that Scrivener is unable to open, although in most cases Scrivener will just use reasonable default values where there is a problem.

It is important to make the distinction between an error that is fatal and one that is not. Any error that results in Scrivener being unable to open the project is considered fatal, whereas other errors are merely considered "invalid" because the program can still open the project by ignoring the problematic element or attribute or using a reasonable default instead. Note that of the XML files Scrivener uses, only an error in the SCRIVX file can result in a "fatal" error.

# *4. The SCRIVX File*

The .scrivx file is one of the most important files inside the .scriv package. It stores the binder structure and all information about binder documents such as meta-data, internal ID and so forth, as well as maintaining project keywords, label and status settings, the auto-complete list and more. It is essentially the workhorse of the app and is what is loaded whenever a .scriv file is opened in order to load the project.

By default, the .scrivx file takes the name of the project, so a project entitled MyProject.scriv would have a MyProject.scrivx file inside it. It doesn't matter if the .scrivx file has a different name, though, as Scrivener will still find it. In this way, other platforms - which don't have support for package file formats - can use the .scrivx file as the file from which to load a project. That is, on platforms other than the Mac, a .scriv package will appear as a regular folder, so the user won't be able to double-click on it to open it. Instead, the user would drill down into this folder and double-click on the .scrivx file to open the project.

Scrivener for the Mac makes a backup of the .scrivx file whenever a project is opened. It zips up the file and stores the compressed backup as "binder.backup" in the /Files directory inside the project package. That way, if the .scrivx is corrupted somehow, Scrivener can look for the backup when it is next opened to load the earlier version (after which it does checks to find any newly created files). It also makes a backup of this file whenever the project is saved, creating a zipped backup entitled binder.autosave in the .scriv/Files folder too.

The .scrivx file has the following root elements (note that by "root elements" I am actually referring to elements one level lower than the true root - the file's root element is entitled **ScrivenerProject**, and that contains these elements):

| | |
|---|---|
| **Binder** | Stores the binder structure, and all information and meta-data pertaining to the individual documents within the project. |
| **Collections** | Stores information about collections (a 2.0-only feature). |
| **Keywords** | Stores the list of keywords associated with this project. Keywords have a title and colour associated with them (colour association is a 2.0-only feature) and are organised in an outline in Scrivener's Keywords HUD, so this is a hierarchical list. |
| **LabelSettings** | Stores the list of labels and their associated colours for the current project. |
| **StatusSettings** | Stores the list of status items for the current project. |
| **CustomMetaDataSettings** | Stores any custom meta-data (outliner columns) that has been set up for this project (2.0-only). |
| **ProjectProperties** | Stores basic information about the project such as the project title and author name (2.1-only). |
| **ProjectTargets** | Stores information about the project targets (draft and session word count/character target). |

| | |
|---|---|
| **TemplateFolderUUID** | Stores the internal UUID of a binder document that is used as a template folder (2.0-only). This folder is assigned a special icon and any subdocuments become available in the "Project > New" menu and can be used as a basis for new documents (e.g. character sheets). |
| **Favorites** | A list of document IDs that will appear at the top of the "Go To", "Move To" menu etc, as assigned by the user. (2.0-only.) |
| **FullScreenBackdrop** | Stores information about the image that is used as the backdrop for full screen, if one is set (2.0-only). The image may be an image document within the project or a file on disk. |
| **ProjectReferences** | Stores a list of project references (as appear in the References pane in the inspector). |
| **AutoCompleteList** | Stores the list of auto-complete words set up for this project. |
| **PrintSettings** | Stores the print settings for this project (paper type, margins etc). |
| **UserProjectData** | Reserved for third-party data (see p.138). |

Elements of the .scrivx XML are described below.

## ScrivenerProject Element

| | |
|---|---|
| **Description:** | This is the root element of a .scrivx file. |
| **Subelements:** | **Binder, Collections, Keywords, LabelSettings, StatusSettings, CustomMetaDataSettings, ProjectProperties, ProjectTargets, TemplateFolderUUID, Favorites, FullScreenBackdrop, ProjectReferences, AutoCompleteList, PrintSettings, UserProjectData** |
| **Attributes:** | **Version, Creator, Device, NewProject, Modified, ModID, SyncID, Checksums** |
| **Number:** | Exactly 1. |
| **Errors:** | A .scrivx file without a ScrivenerProject element is an invalid document. |
| **Value:** | None. |

### Version attribute

| | |
|---|---|
| **Description:** | The **Version** attribute is currently unused but may be used in the future if the file format changes. |
| **Required:** | No. |
| **Value:** | "1.5". (Note: all versions of Scrivener 2.x up to and including 2.5 had a version number of "1.0"; as of 2.6, the version number was updated to "1.5" owing to changes in the file format to accommodate mobile syncing. |

## Creator attribute

**Description:** The **Creator** attribute records the program and version used to save the .scrivx file. This is intended for troubleshooting purposes, so that it is easy to see whether Scrivener created the file (and if so which version of Scrivener) or a third-party tool.

**Required:** Yes.

**Value:** A string that records the program and version number being used to save the file. Scrivener for the Mac uses the format "SCRMAC-[version number]-[build number]. E.g. Scrivener version 2.0.3 (build 7384) would be recorded in this attribute as "SCRMAC-2.0.3-7384".

## Device attribute

**Description:** The **Device** attribute records name of the device that was last used to save the project.

**Required:** Yes.

**Value:** A string identifying the device that last saved the project, e.g. "Keith-Blount-iPad".

## Modified attribute

**Description:** The **Modifed** attribute records the date the project was last saved.

**Required:** Yes.

**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

## ModID attribute

**Description:** The **ModID** attribute stores a UUID that is regenerated every time the desktop version is saved. This attribute should only appear in the desktop .scrivx file, and never in the mobile binder.sync file.

**Required:** Yes on desktop, No on Mobile (the mobile version uses has a **SyncID** instead, and should never have a **ModID** attribute).

**Value:** A string containing a unique identifer. This should be regenerated every time the project is saved, at the same time the **Modifed** is updated, so that the UUID is different for every save.

## SyncID attribute

**Description:** The **SyncID** attribute records the **ModID** of the desktop binder from the time the desktop binder was read. When the mobile version edits a project, it makes a copy of the SCRIVX file in the /Mobile folder entitled "binder.sync". The binder.sync XML

file should remove the **ModID** attribute and replace it with the **SyncID** attribute. Unlike the **ModID** attribue on the desktop, the **SyncID** attribute of the mobile version is *not* regenerated each time the project is saved. It is simply a "frozen" version of the **ModID** attribute from the desktop version as it was at the time the mobile version made its copy of the XML file. The **SyncID** attribute of the mobile binder.sync file and the **ModID** attribute of the desktop .scrivx file can then be compared: if they are the same, we know that the mobile version is newer and no changes have been made to the desktop version since the binder.sync file was created; if they are different, the project must have been edited separately on both mobile and desktop versions.

**Required:** No. Never appears in the desktop version. Only appears in the mobile version when it was opened from a project containing desktop files (it does not appear in projects created on the mobile version that have never been synced with the desktop version).

**Value:** A string containing a unique identifer, copied from the **ModID** attribute of the desktop SCRIVX file.

## Checksums attribute

**Description:** The **Checksums** attribute only appears in the mobile binder.sync file, and never in the desktop SCRIVX file. It contains a unique identifier that matches the file name of the .checksums file inside the /Mobile/Data folder (if there are any files contained in the /Mobile/Data folder).

**Required:** No. Never appears in the desktop version. Only appears in the mobile version when there are files inside the /Mobile/Data folder that have been edited on the mobile version.

**Value:** A string containing a unique identifer that matches the filename of the .checksums file in the /Mobile/Data folder.

## NewProject attribute

**Description:** The **NewProject** attribute is used by Scrivener when opening a project to determine how to load the binder. If **NewProject** is set to "No" and no **Binder** element is found, Scrivener throws an error, as presumably the file is corrupt. However, if **NewProject** is set to "Yes" and no **Binder** element is found, this tells Scrivener to generate the default binder items - the Draft, Research and Trash folders and a single text document inside the Draft folder. This attribute is set to "Yes" when Scrivener creates and loads a new blank project, for instance. It should always be set to "No" (or it should not be present at all) if the **Binder** element exists.

**Required:** No.

**Value:** "Yes" or "No".

## Binder Element

**Description:** The **Binder** element encodes the project binder structure, and thus crucial information about every document in the project. It stores an array of **BinderItem** elements, and these, along with their associated files stored in the .scriv/Files/Docs folder, constitute the basic currency of Scrivener - documents arranged in a structure determined by the user.

**Subelements:** **BinderItem** (see p.43 for a description of the **BinderItem** XML element).

**Attributes:** None.

**Number:** Exactly 1. If not present, then the **NewProject** attribute of the **ScrivenerProject** element must be set to "Yes", otherwise this is an invalid file and Scrivener will refuse to load it. (If **NewProject** is set to "Yes", Scrivener will generate a default empty binder; if not, it expects the **Binder** element to provide the binder.)

**Errors:** The **Binder** element must contain at least three **BinderItem** elements, one of the type "DraftFolder", one of the type "ResearchFolder", and one of the type "TrashFolder". If these three items are not present then the project is not valid and loading will fail.

**Value:** None.

## Collections Element

**Description:** The **Collections** element encodes all the collections of the current project.

**Subelements:** **Collection** (see p.67 for a description of the **Collection** XML element).

**Attributes:** None.

**Number:** Exactly 1. If not present, then Scrivener will create the two required collections (the binder and recent search collections).

**Errors:** The **Collections** element must contain at least two collections, one of the "Binder" type and one of the "RecentSearch" type.

**Value:** None.

## Keywords Element

**Description:** The **Keywords** element encodes all project keywords. Each keyword has an internal ID. When a project keyword is

| | |
|---|---|
| | assigned to a binder document, it is actually the internal ID of the keyword that is associated with the binder document and not the word itself, so that the details of the keyword - its title and colour - can be looked up from the project list and updated in the interface if necessary (if the user renames the keyword, for instance). |
| **Subelements:** | **Keyword** (see p.75 for a description of the **Keyword** XML element). |
| **Attributes:** | None. |
| **Number:** | 1 or 0 if the user has not yet created any keywords for the current project. |
| **Errors:** | None. |
| **Value:** | None. |

## LabelSettings Element

| | |
|---|---|
| **Description:** | The **LabelSettings** element encodes information about the list of labels associated with the current project. Each label has an internal ID. When a label is assigned to a binder document, it is actually the internal ID of the label that is associated with the binder document, so that the details of the label - its title and colour - can be looked up from the project list and updated in the interface if necessary (if the user renames the label, for instance). |
| **Subelements:** | **Title, DefaultLabelID, Labels** |
| **Attributes:** | None. |
| **Number:** | Exactly 1. |
| **Errors:** | None. |
| **Value:** | None. |

## Title Element

| | |
|---|---|
| **Description:** | The **Title** subelement of the **LabelSettings** element encodes the title of the label field the user sees in the inspector and at the top of the label column in the outliner. By default, this is just "Label", but if the user wants to use labels to represent point of view, for instance, the title might be set to "PoV". |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1. |
| **Errors:** | None. |
| **Value:** | A string containing the name to be used for labels in the current project ("Label" by default). |

## DefaultLabelID Element

**Description:** The **DefaultLabelID** encodes the ID of the label that should be assigned to newly created documents in the project ("-1" by default, which represents "No Label").
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** An integer representing the internal ID of the label to be used for new documents.

## Labels Element

**Description:** The **Labels** element encodes the list of labels associated with the current project.
**Subelements:** **Label**
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** None.

## Label Element

**Description:** The **Label** element encodes information about a single label in the current project.
**Subelements:** **ID, Color**
**Attributes:** None.
**Number:** There should be as many **Label** elements inside the **Labels** element as there are labels defined in the current project.
**Errors:** None.
**Value:** A string containing the name of the current label.

### ID attribute

**Description:** The **ID** attribute of the **Label** element encodes the unique internal ID of the current label.
**Required:** Yes.
**Value:** An integer representing the unique internal ID of the current label. "-1" always means "No Label"; all other labels have a value of 0 or above (each one being unique).

### Color attribute

**Description:** The **Color** attribute of the **Label** element encodes the colour to

| | |
|---|---|
| | be used to represent this label. (Label colours may be used to colour the background of index cards, outliner and binder rows and so on.) |
| **Required:** | Yes. If not present, Scrivener will use a random colour. |
| **Value:** | Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green. Note that the colour will be ignored for the "No Label" item (which always has an internal ID of −1), which has no colour associated with it in Scrivener's interface. |

## StatusSettings Element

| | |
|---|---|
| **Description:** | The **StatusSettings** element encodes information about the list of status items associated with the current project. Each status item has an internal ID. When a status item is assigned to a binder document, it is actually the internal ID of the status item that is associated with the binder document, so that the details of the status item - its title and colour - can be looked up from the project list and updated in the interface if necessary (if the user renames the status item, for instance). |
| **Subelements:** | **Title, DefaultStatusID, StatusItems** |
| **Attributes:** | None. |
| **Number:** | Exactly 1. |
| **Errors:** | None. |
| **Value:** | None. |

## Title Element

| | |
|---|---|
| **Description:** | The **Title** subelement of the **StatusSettings** element encodes the title of the status field the user sees in the inspector and at the top of the status column in the outliner. By default, this is just "Status", but the user can change it to anything he or she wishes. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1. |
| **Errors:** | None. |
| **Value:** | A string containing the name to be used to describe status items in the current project ("Status" by default). |

## DefaultStatusID Element

**Description:** The **DefaultStatusID** encodes the ID of the status item that should be assigned to newly created documents in the project ("-1" by default, which represents "No Status").
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** An integer representing the internal ID of the status to be used for new documents.

## StatusItems Element

**Description:** The **StatusItems** element encodes the list of status items associated with the current project.
**Subelements:** **Status**
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** None.

## Status Element

**Description:** The **Status** element encodes information about a single status item in the current project.
**Subelements:** **ID**
**Attributes:** None.
**Number:** There should be as many **Status** elements inside the **StatusItems** element as there are status items defined in the current project.
**Errors:** None.
**Value:** A string containing the name of the current status item.

### ID attribute

**Description:** The **ID** attribute of the **Status** element encodes the unique internal ID of the current status item.
**Required:** Yes.
**Value:** An integer representing the unique internal ID of the current status item. "-1" always means "No Status"; all other status items have a value of 0 or above (each one being unique).

## CustomMetaDataSettings Element

**Description:** The **CustomMetaDataSettings** element defines all custom meta-data fields the user has created for the current project. Custom meta-data fields are essentially custom outliner columns - text-only outliner columns created by the user - although these fields also appear in the inspector.
**Subelements:** **MetaDataField**
**Attributes:** None.
**Number:** 1 or 0 if no custom meta-data fields have been created in the current project.
**Errors:** None.
**Value:** None.

## MetaDataField Element

**Description:** The **MetaDataField** element encodes information about a single custom meta-data field that has been defined for the current project.
**Subelements:** None.
**Attributes:** **ID, Wraps, Color**
**Number:** There should be as many **MetaDataField** elements in the **CustomMetaDataSettings** element as there are custom meta-data fields defined for the current project.
**Errors:** None.
**Value:** A string containing the name of the current meta-data field (the name that will appear in the inspector and at the top of the outliner column).

### ID attribute

**Description:** The **ID** attribute of the **MetaDataField** element encodes a unique string that is used internally to represent the current meta-data field.
**Required:** Yes.
**Value:** A string with no spaces that will be used to represent the meta-data field internally - the user never sees this string. (Scrivener for the Mac uses this string as an identifier for the NSTableView column, for instance.) Scrivener for the Mac creates this string from the first title the user enters for the meta-data field. For example, if the user calls the meta-data field “Location Notes”, Scrivener will use “locationnotes” as the unique ID. If “locationnotes” is already in use, then it will append a number to ensure that it is unique - for instance, “locationnotes1”. Even if the user changes the title “Location Notes” to something else, the “locationnotes” identifier will

remain (unseen by the user) to identify this meta-data field internally, as this is the ID against which **BinderItem** associate values. See the notes on **CustomMetaData** in the **BinderItem** documentation for more information. Note that the following values should **not** be used as ID values, as they may conflict with existing meta-data columns: “title”, “label”, “status”, “createdDate”, “modifiedDate”, “words”, “characters”, “includeInExport”, “pageBreakBefore”, “compileAsIs”, “targetCount”, “targetCountType”, “progress”, “totalWords”, “totalCharacters”, “allKeywords”.

### Wraps attribute

**Description:** The **Wraps** attribute of the **MetaDataField** element determines whether the text of the custom meta-data field should be allowed to wrap to the next line in the outliner and inspector. Normally, the only column in the outliner that allows text wrapping is the title and synopsis column (but only when the synopses are visible). If **Wraps** is set to “Yes”, then the custom column will allow text wrapping too.

**Required:** No. If not present, “No” is assumed.

**Value:** “Yes” or “No”.

### Color attribute

**Description:** The **Color** attribute of the **MetaDataField** element encodes the colour that will be used to draw the text of values in this custom meta-data column.

**Required:** No. If not present, the default text colour is used (usually black).

**Value:** Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green.

## ProjectProperties Element

**Description:** The **ProjectProperties** element encodes basic information about the project such as the project title and author. If it’s not present, or if all its values are blank, sensible defaults are used instead.

**Subelements:** **ProjectTitle, AbbreviatedTitle, FullName, LastName, FirstName.**

**Attributes:** None.

**Number:** Exactly 1. If 0, the project title is taken from the project file name and the user name is taken from Address Book.

**Errors:** None.
**Value:** None.

## ProjectTitle Element

**Description:** The **ProjectTitle** element encodes the project title as set by the user.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If 0, the project title is taken from the project file name (the last path component minus the path extension).
**Errors:** None.
**Value:** None.

## AbbreviatedTitle Element

**Description:** The **AbbreviatedTitle** element encodes the abbreviated project title as set by the user. This is most commonly used in headers and footers via the Scrivener 2_6 File Format tag, where using the full project title would be undesirable.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If 0, the project title is taken from the project file name (the last path component minus the path extension).
**Errors:** None.
**Value:** None.

## FullName Element

**Description:** The **FullName** element encodes the name of the author, as set by the user.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If 0, the the user's name is used, taken from the system's Address Book.
**Errors:** None.
**Value:** None.

## LastName Element

**Description:** The **LastName** element encodes the last name of the author, as set by the user.

**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If 0, the the user's last name is used, taken from the system's Address Book.
**Errors:** None.
**Value:** None.

## FirstName Element

**Description:** The **FirstName** element encodes the first name of the author, as set by the user.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If 0, the the user's first name is used, taken from the system's Address Book.
**Errors:** None.
**Value:** None.

## ProjectTargets Element

**Description:** The **ProjectTargets** element encodes the word and character count targets for the current project.
**Subelements:** **DraftTarget, SessionTarget, PreviousSession.**
**Attributes:** **Notify**
**Number:** Exactly 1. If 0, no targets are set.
**Errors:** None.
**Value:** None.

### Notify attribute

**Description:** The **Notify** attribute of the **ProjectTargets** element encodes whether Scrivener should show some sort of notification when the user meets the specified targets. Scrivener for the Mac uses the Growl framework to show a pop-up window or play a user-defined sound if **Notify** is set to "Yes".
**Required:** No. If no **Notify** attribute is provided, "No" is assumed.
**Value:** "Yes" or "No".

## DraftTarget Element

**Description:** The **DraftTarget** element encodes the word or character count target for the Draft of the current project.
**Subelements:** None.
**Attributes:** **Type, CountIncludedOnly, CurrentCompileGroupOnly, Deadline,**

**IgnoreDeadline**

**Number:** Exactly 1. If 0, no Draft target is set.
**Errors:** None.
**Value:** An integer specifying the target for the Draft in words or characters (as determined by the **Type** attribute). For instance, <DraftTarget Type="Words">5000</Target> sets a target of 5,000 words for the current document, whereas <DraftTarget Type="Characters">5000</Target> sets a target of 5,000 characters.

## Type attribute

**Description:** The **Type** attribute of the **DraftTarget** element encodes the target type - i.e. Whether the target should be measured in words or characters.
**Required:** Yes. If no **Type** is provided, the target type is assumed to be "Words".
**Value:** "Words", "Characters" or "Pages".

## CountIncludedOnly attribute

**Description:** The **CountIncludedOnly** attribute of the **DraftTarget** element encodes whether the Draft target counts the text of all documents inside the Draft folder or only the ones with "Include in Compile" set to YES.
**Required:** No. If not present, assume "No".
**Value:** "Yes" or "No".

## CurrentCompileGroupOnly attribute

**Description:** The **CurrentCompileGroupOnly** attribute of the **DraftTarget** element encodes whether or not the Draft target counts only the documents inside the group currently set as the compile group in the Compile sheet. The user can choose not to compile the whole Draft folder but only a subfolder within the Draft. If this attribute is set to "Yes", then only the contents of the subfolder selected for Compile will be counted when calculating how close the user is to reaching his or her targets. (Obviously, if the Draft folder is set as the compile group then this setting has no effect.)
**Required:** No. If not present, assume "No".
**Value:** "Yes" or "No".

## Deadline attribute

**Description:** The **Deadline** attribute of the **DraftTarget** element encodes whether the Draft target counts the text of all documents inside the Draft folder or only the ones with "Include in Compile" set to YES.
**Required:** No. If not present, assume today's date and set **IgnoreDeadline**

to “Yes”.

**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

### IgnoreDeadline attribute

**Description:** The **IgnoreDeadline** attribute of the **DraftTarget** element encodes whether or not the deadline encoded in **Deadline** is used.

**Required:** No. If not present, assume “Yes” (i.e. ignore the deadline).

**Value:** “Yes” or “No”.

### ShowOverrun attribute

**Description:** The **ShowOverrun** attribute of the **DraftTarget** element encodes whether target overruns should be shown in the interface.

**Required:** No. If no **ShowOverrun** attribute is provided, “No” is assumed.

**Value:** “Yes” or “No”.

### Buffer attribute

**Description:** The **Buffer** attribute of the Draft**Target** element encodes how much over the target the user can go before Scrivener starts showing the overrun indicator.

**Required:** No. If no **Buffer** attribute is provided, a 0 buffer is assumed.

**Value:** An integer representing the target overrun buffer.

## SessionTarget Element

**Description:** The **SessionTarget** element encodes the word or character count target for the current session. (At the time of writing, on Scrivener for the Mac the “session” refers to a single launch of a project - so the session word or character count gets reset when the user closes and reopens a project. This may change in the future to refer to a single day of writing.)

**Subelements:** None.

**Attributes:** **Type, CountDraftOnly, AllowNegatives, ResetType, DeterminedFromDeadline, ExcludeWeekends**

**Number:** Exactly 1. If 0, no session target is set.

**Errors:** None.

**Value:** An integer specifying the target for the current session in words or characters (as determined by the **Type** attribute). For instance, <SessionTarget Type=“Words”>5000</Target> sets a target of 5,000 words for the current document, whereas <SessionTarget Type=“Characters”>5000</Target> sets a target of 5,000 characters.

### Type attribute

**Description:** The **Type** attribute of the **SessionTarget** element encodes the target type - i.e. Whether the target should be measured in words or characters.

**Required:** Yes. If no **Type** is provided, the target type is assumed to be "Words".

**Value:** "Words", "Characters" or "Pages".

### CountDraftOnly attribute

**Description:** The **CountDraftOnly** attribute of the **SessionTarget** element encodes whether or not the session target should only count the contents of text contained inside the Draft folder. If set to "No", the session target will increment when the user types anything, anywhere in the project.

**Required:** No. If not present, assume "No".

**Value:** "Yes" or "No".

### AllowNegatives attribute

**Description:** The **AllowNegatives** attribute of the **SessionTarget** element encodes whether or not the session target can show negative numbers (i.e. when the user has deleted more text than he or she has added during a session).

**Required:** No. If not present, assume "No".

**Value:** "Yes" or "No".

### ResetType attribute

**Description:** The **ResetType** attribute of the **SessionTarget** element encodes when the session word and character counts should get reset. There are currently four options: 1) The session counts get automatically reset at midnight, but are remembered if you close and reopen the project on the same day; 2) The session counts are not remembered between sessions, so get reset every time the project is closed and reopened; 3) The session counts get reset when you close and reopen the project after midnight (this is the same as getting reset at midnight, except that the counts don't automatically reset until you close and reopen the project); 4) The session counts are persistent and are never automaically reset, leaving the user to handle resets manually (by clicking on the "Reset" button).

**Required:** No. If not present, assume "Midnight".

**Value:** "Midnight", "ProjectClose", "NextDay" or "Never".

### DeterminedFromDeadline attribute

**Description:** The **DeterminedFromDeadline** attribute of the **SessionTarget** element encodes whether or not the session target should be

calculated automatically based on the **Deadline** and target for the draft. For instance, if this is "Yes" and the draft target is 50,000 words, the deadline date is set for two days' time, and the user has already written 40,000 words in the draft, then the session target would automatically be calculated to be 5,000 words (because there are 10,000 words left to go and two days until the deadline). If this is set to "No", then the user can set the session target manually.

**Required:** No. If not present, assume "No".

**Value:** "Yes" or "No".

### WritingDays attribute

**Description:** The **WritingDays** attribute of the **SessionTarget** element encodes which weekdays should be included when automatically calculating the session target from the deadline (so that the deadline is based only on the days on which the user actually writes).

**Required:** No. If not present, assume that the user writes on every day of the week.

**Value:** A range of numbers indicating which days of the week should be included, where Sunday is 1 and Saturday is 7. The numbers should be separated by a comma and a space or, if sequential, a range can be indicated with a hyphen. For instance, "1, 3-5, 7" would indicate Sunday, Tuesday, Wednesday, Thursday and Saturday.

### CanWriteOnDeadlineDate attribute

**Description:** The **CanWriteOnDeadlineDate** attribute of the **SessionTarget** element encodes whether the day of the actual deadline should be included as a valid writing day when automatically calculating the session target from the deadline. (Normally you wouldn't want to plan on still be writing on the day that something has to be handed in, but sometimes, such as with NaNoWriMo where you can write up to midnight, this is desirable.)

**Required:** No. If not present, assume "No".

**Value:** "Yes" or "No".

## PreviousSession Element

**Description:** The **PreviousSession** element encodes the number of words and characters written when the project was last opened, along with the date on which the project was saved.

**Subelements:** None.

**Attributes:** **Words, Characters, Date**

**Number:** Exactly 1. If 0, the current session word and character counts are just set to 0.
**Errors:** None.
**Value:** None.

### Date attribute

**Description:** The **Date** attribute of the **PreviousSession** element encodes the date of the previous session. Scrivener uses it when opening the project to determine whether the project is being opened on the same day as the previous writing session. If so, the **Words** and **Characters** attributes are used to restore the word and character counts, otherwise they are ignored.
**Required:** Yes. If no **Words** attribute is provided, set the session word count to 0.
**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

### Words attribute

**Description:** The **Words** attribute of the **PreviousSession** element encodes the number of words written during the previous session.
**Required:** Yes. If no **Words** attribute is provided, set the session word count to 0.
**Value:** An integer specifying the number of words written during the previous session.

### Characters attribute

**Description:** The **Characters** attribute of the **PreviousSession** element encodes the number of characters written during the previous session.
**Required:** Yes. If no **Characters** attribute is provided, set the session character count to 0.
**Value:** An integer specifying the number of characters written during the previous session.

## TemplateFolderUUID Element

**Description:** The **TemplateFolderUUID** element encodes the internal UUID of a folder that is set to act as the templates folder. (The subdocuments of the templates folder can be used as the basis for creating new documents, and are listed in from the New From Templates menu.)
**Subelements:** None.
**Attributes:** None.

**Number:** 1 or 0 if no template folder is set for the current project.
**Errors:** None.
**Value:** A string representing the internal UUID (as set in the **UUID** attribute of **BinderItem**) of a folder that is set to act as the templates folder.

## Favorites Element

**Description:** The **Favorites** element encodes a list of internal IDs (as encoded by **BinderItem** elements) representing binder documents that are saved as project "favourites". Favourite documents appear at the top of all menus that display the entire binder hierarchy. For instance, the "Go To", "Move To" and "Scrivener Link" menus all show a list of every binder document in the project (the menus are hierarchical, matching the binder hiearchy). Any documents included in the "Favorites" list also appear at the very top of such menus so that the user can access them quickly.
**Subelements:** **BinderID**.
**Attributes:** None.
**Number:** 1 or 0 if there are no favourite documents set for the current project.
**Errors:** None.
**Value:** None.

## BinderID Element

**Description:** When appearing as a sub-element of the **Favorites** element, the **BinderID** element encodes the internal ID of a single binder document included in the favourites list.
**Subelements:** None.
**Attributes:** None.
**Number:** There should be as many **BinderID** elements inside the **Favorites** element as there are favourite documents for this project.
**Errors:** None.
**Value:** An integer representing the internal ID (as set in the **ID** attribute of **BinderItem**) of a binder document that is included in the list of favourites for the current project.

## FullScreenBackdrop Element

**Description:** The **FullScreenBackdrop** element encodes information about

the background image that should be used in full screen mode, if any.

**Subelements:** None.

**Attributes:** **Type**

**Number:** 1 or 0 if full screen mode doesn't have a background image set in the current project.

**Errors:** None.

**Value:** If the **Type** attribute is set to "BinderID", then the value should be an integer repesenting the internal ID of an image document in the project (as encoded in the **ID** attribute of a **BinderItem** element of the "Image" type). If **Type** is set to "Path", the value should be a string containing the full path to the image on disk that should be used for the full screen background.

### Type attribute

**Description:** The **Type** attribute of the **FullScreenBackdrop** element encodes whether the image used for the full screen background has been imported into the project as an image document or exists separately on disk.

**Required:** Yes. If not present, no full screen backdrop will be used.

**Value:** "BinderID" if the full screen background uses the image from an image document inside the project, or "Path" if it uses an external image located elsewhere on the user's hard drive.

### BookmarkData attribute

**Description:** The **BookmarkData** attribute of the **FullScreenBackdrop** element encodes the necessary authorisation information for the Mac version of Scrivener to access the image file when running in the sandbox. This is only needed for the Mac App Store version of Scrivener, and is only present when the "Path" type is being used. Other versions of Scrivener should preserve this data if present, but should not create it if not.

**Required:** Yes for the Mac sandboxed version, no for other versions.

**Value:** A hexidecimal string encoding an NSData object generated from a security-scoped URL.

## ProjectReferences Element

**Description:** The **ProjectReferences** element encodes any references associated with the whole project. Project references are accessed via the inspector and may contain links to other documents in the project or to external files. Project references are exactly the same as document references (see the **References** element of **BinderItem**), except that whereas document references can only be viewed when the document

| | |
|---|---|
| | with which they are associated is loaded into an editor, project references can be viewed when any document is visible - the user can freely switch between project and document references in the inspector. |
| **Subelements:** | **Reference** |
| **Attributes:** | None. |
| **Number:** | 1 or 0 if the current project contains no project references. |
| **Errors:** | None. |
| **Value:** | None. |

## Reference Element

| | |
|---|---|
| **Description:** | The **Reference** element encodes a single reference that is associated with the current project. |
| **Subelements:** | None. |
| **Attributes:** | **BinderID**, **Destination**. (Note that **BinderID** is used for internal links - links to other documents within the project - and **Destination** is used for external links - web page URLs or external files - and so are mutally exclusive.) |
| **Number:** | There should be as many **Reference** elements as there are references associated with the current project. |
| **Errors:** | None. |
| **Value:** | A string containing the title of the current reference (the user can give references any title). |

### BinderID attribute

| | |
|---|---|
| **Description:** | The **BinderID** attribute of the **Reference** element encodes the internal ID of a binder document within the current project. |
| **Required:** | *Either* the **BinderID** *or* the **Destination** attribute must be present. If neither is present, the **Reference** element refers to nothing and is ignored. |
| **Value:** | An integer representing the internal ID of a document in the current project. |

### Destination attribute

| | |
|---|---|
| **Description:** | The **Destination** attribute of the **Reference** element encodes the URL to an external file or web page. |
| **Required:** | *Either* the **BinderID** *or* the **Destination** attribute must be present. If neither is present, the **Reference** element refers to nothing and is ignored. |
| **Value:** | A string representing the URL to an external file or web page. |

## AutoCompleteList Element

**Description:** The **AutoCompleteList** element encodes a custom auto-complete list the user has created for the current project.
**Subelements:** **Completion**
**Attributes:** None.
**Number:** 1 or 0 if no custom auto-complete list has been created for the current project.
**Errors:** None.
**Value:** None.

## Completion Element

**Description:** The **Completion** element encodes a single word or phrase which has been added to the custom auto-complete list of the current project.
**Subelements:** None.
**Attributes:** **Scope**
**Number:** There should be as many **Completion** elements inside the **AutoCompleteList** element as there are custom auto-complete words and phrases in the current project.
**Errors:** None.
**Value:** A string containing a word or phrase the user has set to be auto-completed in the current project.

### Scope attribute

**Description:** The **Scope** attribute of the **Completion** element encodes where in the text the completion should be used.
**Required:** No. If not present, −2 is assumed (the completion will be available everywhere).
**Value:** An integer that determines where the completion should be used in the project, as follows: a value of −2 indicates that the completion can be used anywhere in the project, in any script element and in non-script documents; a value of −1 indicates that it can be used only in non-script documents; providing the index of one of the project's script elements indicates that the completion can only be used in that script element. For instance, in a project with the script elements, "Scene Heading", "Action", "Dialogue", "Character", "Parentheses", "Transition", and "Shot", a value of 0 would indicate that the completion can only be used in the Scene Heading element, and a value of 3 would indicate that it can only be used in the Character element.

## PrintSettings Element

**Description:** The **PrintSettings** element encodes basic information about the current print settings of the project, such as paper size and margins.
**Subelements:** None.
**Attributes:** **PaperSize, LeftMargin, RightMargin, TopMargin, BottomMargin, PaperType, Orientation, HorizontalPagination, VerticalPagination, ScaleFactor, HorizontallyCentered, VerticallyCentered, Collates, ReverseOrder, PagesAcross, PagesDown**
**Number:** 1 or 0 if Scrivener should just use the default print settings for the current system.
**Errors:** None.
**Value:** None.

### PaperSize attribute

**Description:** The **PaperSize** attribute of the **PrintSettings** element encodes the current paper size to be used for printing.
**Required:** No. The default paper size will be used if not present.
**Value:** Two floating point numbers separated by a comma with no space. The first represents the paper width and the second the paper height, measured in points.

### LeftMargin attribute

**Description:** The **LeftMargin** attribute of the **PrintSettings** element encodes the left margin to be set when printing.
**Required:** No. The default margin will be used if not present.
**Value:** A single floating point number representing the margin as measured in points.

### RightMargin attribute

**Description:** The **RightMargin** attribute of the **PrintSettings** element encodes the right margin to be set when printing.
**Required:** No. The default margin will be used if not present.
**Value:** A single floating point number representing the margin as measured in points.

### TopMargin attribute

**Description:** The **TopMargin** attribute of the **PrintSettings** element encodes the top margin to be set when printing.
**Required:** No. The default margin will be used if not present.
**Value:** A single floating point number representing the margin as measured in points.

### BottomMargin attribute

**Description:** The **BottomMargin** attribute of the **PrintSettings** element encodes the bottom margin to be set when printing.
**Required:** No. The default margin will be used if not present.
**Value:** A single floating point number representing the margin as measured in points.

### PaperType attribute

**Description:** The **PaperType** attribute of the **PrintSettings** element encodes the name of the paper type to be used when printing, e.g. “iso-a4” or “Letter”.
**Required:** No. Defaults to “Letter” if not present.
**Value:** A string representing the name of the paper type to use when printing.

### Orientation attribute

**Description:** The **Orientation** attribute of the **PrintSettings** element encodes whether printing should use landscape or portrait mode.
**Required:** No. Defaults to “Portrait” if not present.
**Value:** “Landscape” or “Portrait”.

### HorizontalPagination attribute

**Description:** The **HorizontalPagination** attribute of the **PrintSettings** element encodes the horizontal pagination type.
**Required:** No. Defaults to “Auto” if not present.
**Value:** “Clip” if an image to be printed should be clipped horizontally, “Fit” if it should be scaled to fit horizontally, and “Auto” if it should be divided across different pages.

### VerticalPagination attribute

**Description:** The **VerticalPagination** attribute of the **PrintSettings** element encodes the vertical pagination type.
**Required:** No. Defaults to “Auto” if not present.
**Value:** “Clip” if an image to be printed should be clipped vertically, “Fit” if it should be scaled to fit vertically, and “Auto” if it should be divided across different pages.

### ScaleFactor attribute

**Description:** The **ScaleFactor** attribute of the **PrintSettings** element encodes the scale (or zoom) factor to use for printing.
**Required:** No. Defaults to “1.0” if not present.
**Value:** A floating point number representing the current scale factor, where 1.0 is the original size (so that 2.0 prints at double-size and 0.5 prints at half-size).

### HorizontallyCentered attribute

**Description:** The **HorizontallyCentered** attribute of the **PrintSettings** element determines whether an image being printed should be centred horizontally on the page.
**Required:** No. Defaults to "No" if not present.
**Value:** "Yes" or "No".

### VerticallyCentered attribute

**Description:** The **VerticallyCentered** attribute of the **PrintSettings** element determines whether an image being printed should be centred vertically on the page.
**Required:** No. Defaults to "No" if not present.
**Value:** "Yes" or "No".

### Collates attribute

**Description:** The **Collates** attribute of the **PrintSettings** element determines whether printing should be collated.
**Required:** No. Defaults to "Yes" if not present.
**Value:** "Yes" or "No".

### ReverseOrder attribute

**Description:** The **ReverseOrder** attribute of the **PrintSettings** element determines whether pages should be printed in reverse order.
**Required:** No. Defaults to "No" if not present.
**Value:** "Yes" or "No".

### PrintHeaderAndFooter attribute

**Description:** The **PrintHeaderAndFooter** attribute of the **PrintSettings** element determines whether a header and footer should be printed on the page. Note, however, that Scrivener for the Mac does not currently use this setting anywhere. This encodes a default OS setting, but Scrivener overrides it with its own settings in the Compile draft sheet (whose settings are saved in .scriv/Settings/compile.plist and in its Page Setup sheet (whose settings are saved in .scriv/Settings/pagesetup.plist).
**Required:** No. Defaults to "No" if not present.
**Value:** "Yes" or "No".

### PagesAcross attribute

**Description:** The **PagesAcross** attribute of the **PrintSettings** element determines how many logical pages should be tiled horizontally across a single sheet of paper (so that more than one page of a Scrivener document can be printed on a single physical page).
**Required:** No. Defaults to "1" if not present.
**Value:** An integer representing the number of document pages that

should be tiled across a single printed page.

### PagesDown attribute

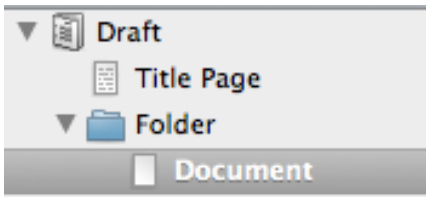
**Description:** The **PagesDown** attribute of the **PrintSettings** element determines how many logical pages should be tiled vertically down a single sheet of paper (so that more than one page of a Scrivener document can be printed on a single physical page).
**Required:** No. Defaults to "1" if not present.
**Value:** An integer representing the number of document pages that should be tiled down a single printed page.

## UserProjectData Element

**Description:** The **UserProjectData** element encodes and preserves data added by third-party applications (or added directly by the user). See p.138 for details.
**Subelements:** Any custom elements added by users or third-party applications.
**Attributes:** None.
**Number:** 1 or 0 if no custom data has been added.
**Errors:** None.
**Value:** None.

## *4.1. The BinderItem Element*

The **BinderItem** XML element constitutes the core of Scrivener's XML file format. The **BinderItem** element encodes a single binder document, recording whether it is a text document, folder, PDF document, whether it is set to be included in Compile, any keywords or references associated with it and so on. A **BinderItem** element can contain any number of other **BinderItem** elements inside its **Children** subelement, and so in this way the entire binder structure can be encoded using the basic unit of **BinderItem** XML elements. For instance, the Draft folder in the binder will be encoded as a **BinderItem** XML element that contains any number of other **BinderItem** elements of the "Text" or "Folder" type, representing each document inside the Draft folder in the binder. In turn, any of the these text or folder subelements may have subelements of their own (contained in their own **Children** elements). To clarify, consider the following structure:




A simplified version of the XML structure would look like this:

```
<BinderItem Type="DraftFolder">          // Draft folder starts
    <Title>Draft</Title>
    <Children>                           // Subdocuments of the Draft folder
        <BinderItem Type="Text">         // A single text doc entitled "Title Page" (inside Draft)
            <Title>Title Page</Title>
        </BinderItem>
        <BinderItem Type="Folder">       // A folder inside the Draft folder
            <Title>Folder</Title>
            <Children>                   // Subdocuments of the folder
                <BinderItem Type="Text"> // A single text doc inside the folder entitled "Document"
                    <Title>Document</Title>
                </BinderItem>
            </Children>                  // End of the folder's subdocuments
        </BinderItem>
    </Children>                          // End of Draft folder subdocuments
</BinderItem>                            // Draft folder ends
```

Elements of the **BinderItem** XML element are as follows:

### BinderItem Element

| | |
|---|---|
| **Description:** | This is the root element for representing a single binder document. |
| **Subelements:** | **Title, MetaData, TextSettings, MediaSettings, CorkboardAndOutliner, Keywords, References, UserData, Children** |
| **Attributes:** | **ID, Type, Created, Modified** |
| **Number:** | There should be as many **BinderItem** elements within the **Binder** |

element of the SCRIVX file as there are binder documents in the root of the binder. Likewise, there should be as many child **BinderItem** elements inside the **Children** element of each **BinderItem** as there are subdocuments in the binder for that document.

**Errors:** A **Binder** element with no **BinderItem** elements is invalid. There should always be at least three **BinderItem** elements inside the **Binder** element, one of the "DraftFolder" type, one of the "ResearchFolder" type, and one of the "TrashFolder" type. If a **BinderItem** element of each of these three types is not present in the **Binder** element, then the project is invalid.

**Value:** None.

## ID attribute

**Description:** The **ID** attribute of the **BinderItem** element encodes the internal binder ID of the document represented by the current **BinderItem** element. This internal ID is used by Scrivener to keep track of the document and is also used as the file name for saving the document to disk. For example, a text document with an ID of 37 would have its text saved as 37.rtf inside the .scriv/Files/Docs folder; a PDF file with an ID of 19 would have its associated file stored as 19.pdf inside the .scriv/Files/Docs folder; the notes of a document with an internal ID of 138 would be saved as 138_notes.rtf inside the .scriv/Files/Docs folder. The **ID** attribute is thus essential for the proper working of a Scrivener project.

**Required:** Yes in desktop version. Without an **ID** attribute, a **BinderItem** is invalid on the desktop version. On the mobile binder.sync file, there should only be an **ID** attribute if the **BinderItem** also exists in the desktop SCRIVX file. Binder items created anew in the mobile version are not assigned an **ID** attribute (only a **UUID** attribute) - the desktop version will assign an **ID** when it syncs with the mobile version.

**Value:** An integer representing the internal ID of the document. Note that the Draft folder must always have an ID of 0, the Research folder must always have an ID of 1, and the Trash folder must always have an ID of 2. IDs of other documents added to the project thus start at 3.

## UUID attribute

**Description:** The **UUID** attribute encodes an internal unique identifier for the document represented by the current **BinderItem** element. This internal UUID is used by Scrivener to keep track of the document and is also used in the file structure for saving the document to disk on the mobile version.

**Required:** Yes. Without an **UUID** attribute, a **BinderItem** is invalid.

**Value:** A string containing a unique identifier.

### Type attribute

**Description:** The **Type** attribute of the **BinderItem** element encodes the document type - i.e. Whether it is a folder, a text item, a PDF document etc.

**Required:** Yes. If no **Type** is provided, the **BinderItem** is assumed to be a text document.

**Value:** One of the following values: “DraftFolder”, “ResearchFolder”, “TrashFolder”, “Folder”, “Text”, “Image”, “PDF”, “Media” (movie and sound files, which are both played using the QuickTime framework), “WebArchive”, “Other”. (Note that “Other” is used as the type for any unsupported document type that is imported into Scrivener - Scrivener 2.0 can import any file type, even files it cannot display in its own editor. In this way a user can store all files associated with his or her project in one .scriv file and open any unsupported file types in an external editor using Scrivener’s “Open in External Editor” feature.)

### Created attribute

**Description:** The **Created** attribute of the **BinderItem** element encodes the created date of the binder document (this date is displayed in Scrivener’s inspector, for instance).

**Required:** Yes. If the created date is not present, today’s date will be used.

**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

### Modified attribute

**Description:** The **Modified** attribute of the **BinderItem** element encodes the date the binder document was last modified (i.e. edited) - this date is displayed in Scrivener’s inspector, for instance.

**Required:** Yes. If the modified date is not present, today’s date will be used.

**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

## Title Element

**Description:** The **Title** element encodes the title of the binder document.

**Subelements:** None.

**Attributes:** None.
**Number:** Exactly 1. If not present, "Untitled" will be used.
**Errors:** None.
**Value:** A string representing the title of the current binder document.

## GroupMode Element

**Description:** The **GroupMode** element encodes the view mode a group document should open in (corkboard, outliner etc). This element is usually not present, as groups are usually opened in whichever view mode was last used for a group. This setting allows users to set a particular group mode for particular documents, however.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If not present, the default view mode will be used.
**Errors:** None.
**Value:** "Corkboard", "Outliner", "Scrivenings", "Document" or "Default". This element can be omitted altogether instead of writing it out with "Default", though.

## MetaData Element

**Description:** The **MetaData** element encodes meta-data associated with the binder document such as label, status, "page break before" etc (meta-data for each document is displayed in Scrivener's inspector).
**Subelements:** **LabelID, StatusID, PageBreakBefore, IncludeInCompile, CompileAsIs, FileExtension, NotesTextSelection, ShowSynopsisImage, IndexCardImageFileExtension, IgnoresTitlePrefixAndSuffix, IconFile, PreferredExternalApplication, CustomMetaData**
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** None.

## LabelID Element

**Description:** The **LabelID** element encodes the internal ID representing the label that has been applied to the binder document encoded by the current **BinderItem**. (Note that the name and colour of the label associated with this ID - what the user sees - is encoded in the **LabelSettings** element of the .scrivx file.)

**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, the LabelID is assumed to be −1, which represents “No Label”.
**Errors:** None.
**Value:** An integer representing the internal ID of the label applied to this document, as defined in the **LabelSettings** element of the SCRIVX file.

## StatusID Element

**Description:** The **StatusID** element encodes the internal ID representing the status that has been applied to the binder document encoded by the current **BinderItem**. (Note that the name associated with this ID - what the user sees - is encoded in the **StatusSettings** element of the .scrivx file.)
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, the StatusID is assumed to be −1, which represents “No Status”.
**Errors:** None.
**Value:** An integer representing the internal ID of the status applied to this document, as defined in the **StatusSettings** element of the main SCRIVX file.

## PageBreakBefore Element

**Description:** The **PageBreakBefore** element encodes whether or not the binder document encoded by the current **BinderItem** is set to have a page break inserted before it when compiled. This information can be set via Scrivener’s inspector or Compile sheet. (Note that this setting is only used for documents contained inside the Draft folder.)
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, **PageBreakBefore** is assumed to be “No”.
**Errors:** None.
**Value:** “Yes” or “No”.

## IncludeInCompile Element

**Description:** The **IncludeInCompile** element encodes whether or not the binder document is set to be included when compiling the draft

into one large document via the Compile feature. This information can be set via Scrivener's inspector or Compile sheet. (Note that this setting is only used for documents stored inside the Draft folder.)

**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, **IncludeInCompile** is assumed to be "No".
**Errors:** None.
**Value:** "Yes" or "No".

## CompileAsIs Element

**Description:** The **CompileAsIs** element encodes whether or not the binder document should ignore any text formatting applied via the Compile feature. If set to "Yes", the document will appear in the compiled draft document exactly as it does in Scrivener's editor, and only its text will be included (i.e. it will ignore the Compile settings and include only the text regardless - no title will be added and the text will be included even if the user has set text not to be included for documents of this type and level); if set to "No", it will use any formatting the user applies at the Compile stage, and include the elements specified by the user (title, notes etc). This information can be set via Scrivener's inspector or Compile sheet. (Note that this setting is only used for documents stored inside the Draft folder.)

**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, **CompileAsIs** is assumed to be "No".
**Errors:** None.
**Value:** "Yes" or "No".

## FileExtension Element

**Description:** The **FileExtension** element encodes the file extension of the underlying file associated with the binder document. This is used by media and image files, which can have a range of possible extensions. When reading a file from disk, Scrivener uses the internal ID of the document (encoded by the **ID** attribute of the **BinderItem** element) and appends a path extension, then looks inside the .scriv/Files/Docs folder to find the file. For instance, to find a text file with the internal ID of 77, it would look inside the .scriv/Files/Docs folder for a file entitled 77.rtf (because text files are always saved as RTF files). When looking for a movie file, though, the movie file

| | |
|---|---|
| | may have any valid movie extension - e.g., .mov, .mp4 etc. Likewise, an image file might be a .jpg file, a .tif file, a .png file etc. For such files, Scrivener uses the **FileExtension** element of the **MetaData** element to determine the path of the file. For example, for an image file with an internal ID of 154 and a **FileExtension** of "jpg", Scrivener would look inside the .scriv/Files/Docs folder for a file entitled 154.jpg. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0. This element must be present for media and image files. (For text files, Scrivener will automatically add "rtf", for PDF files it will automatically add "pdf", and for web files it will automatically add "webarchive". Note that it is valid to include the file extension for any file type though, so there is nothing wrong with including "rtf" as the extension for text files, even though it will not be used.) |
| **Errors:** | None. |
| **Value:** | A string representing the file extension of the file associated with the current binder document, e.g. "jpg", "mov" etc. |

## NotesTextSelection Element

| | |
|---|---|
| **Description:** | The **NotesTextSelection** element encodes the range of currently selected text for the notes associated with the document encoded by the current **BinderItem** element. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0. If not present, a range of 0,0 is assumed (i.e. no selection with the cursor at the start of the text). |
| **Errors:** | None. |
| **Value:** | Two integers separated by a comma (with no space). The first number indicates the index of the character at which the selection starts, and the second number describes the number of characters across which the selection applies. So, for instance, if the word "Science" is currently selected and the "S" of "Science" is the tenth letter in the text, then the range would be 9,7 (because the tenth letter is at index 9 and the word "Science" contains 7 characters). Note that if there is no selection, the second integer should be 0. For instance, a value of 1,0 indicates that the cursor is currently inserted before the second character. |

## ShowSynopsisImage Element

| | |
|---|---|
| **Description:** | The **ShowSynopsisImage** element encodes whether or not the binder document shows an image in place of the synopsis text in the inspector and corkboard. (Scrivener 2.0 allows the user to freely switch between showing an image or text in an index card.) |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0. If not present, **ShowSynopsisImage** is assumed to be "No". |
| **Errors:** | None. |
| **Value:** | "Yes" or "No". |

## IndexCardImageFileExtension Element

| | |
|---|---|
| **Description:** | The **IndexCardImageFileExtension** element encodes the file extension of the image that can be shown in place of the synopsis in the index card associated with the binder document. These index card images are saved in the .scriv/Files/Docs folder with the following file name format: [BinderID]_icImage.[imageExtension}. For example, if a document with the internal ID of 125 has **ShowSynopsisImage** set to "Yes" and an **IndexCardImageFileExtension** value of "png", then Scrivener will look inside the .scriv/Files/Docs folder for a file entitled 125_icImage.png to use as the index card image (hence "icImage" - "index card image"). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 if the current binder document has an index card image associated with it. |
| **Errors:** | None. If not present, the binder document will not show an index card image. |
| **Value:** | A string representing the file extension of the index card image, e.g. "jpg". |

## IgnoresTitlePrefixAndSuffix Element

| | |
|---|---|
| **Description:** | The **IgnoresTitlePrefixAndSuffix** element encodes whether or not the binder document should ignore any title prefix and suffix settings set in Compile. In Scrivener 2.0, it is possible to set up the Compile process so that a binder document's title is automatically prefixed and suffixed by custom text set by the user. For instance, the user may choose to have all folder titles |

prefixed with “Chapter 1: ”, so that upon compile, folders entitled “Foo” and “Bar” become “Chapter 1: Foo” and “Chapter 2: Bar”. The user can set individual documents to ignore these prefix and suffix settings, though, by selecting “Ignore Title Prefix and Suffix” in the Compile sheet - for instance, the user may not wish the “Prologue” or “Epilogue” to have any prefix. The **IgnoreTitlePrefixAndSuffix** element encodes this setting.

**Subelements:** None.
**Attributes:** None.
**Number:** 1 if 0. If not present, “No” is assumed.
**Errors:** None.
**Value:** “Yes” or “No”.

## IconFileName Element

**Description:** The **IconFileName** element encodes the name of any custom icon associated with the binder document. Scrivener 2.0 allows the user to apply custom icons to any document that will be used instead of the default document and folder icons. These icon image files can be stored inside the .scriv project itself (in the .scriv/Icons directory) or in a folder in Application Support (on the Mac). The **IconFileName** element encodes only the name of the icon file, e.g. “MyImage.jpg”. Scrivener first looks for this image in the .scriv/Icons folder, and if it isn’t found there, then looks in the Application Support folder.

**Subelements:** None.
**Attributes:** None.
**Number:** 1 if 0. If not present, the document has no custom icon associated with it.
**Errors:** None.
**Value:** A string representing the file name of an icon that is stored inside the .scriv/Icons folder or the Application Support folder (on the Mac), e.g. “MyIcon.tif”. (Note that icon images should ideally be 16x16 pixels in size.)

## PreferredExternalApplication Element

**Description:** The **PreferredExternalApplication** element encodes the path to, or name of, the application that will be used to open the binder document if the user selects “Open in External Application”. (Note that this only applies to media files such as PDF files, images, movie and sound files etc, and cannot be used with text files inside Scrivener.) If **PreferredExternalApplication** is not

set, then when using "Open in External Application", Scrivener will open the file in the default application associated with the file (e.g. Preview for PDF files on the Mac).

**Subelements:** None.

**Attributes:** None.

**Number:** 1 if 0. If not present, the default application associated with the file type will be used to open the file, as determined by the system. This element is ignored for text and folder files which cannot be opened in external applications.

**Errors:** None.

**Value:** A string representing the path to or name of an external application that should be used as the preferred application for opening the current binder document when using "Open in External Editor". Thus, if the preferred external application for a PDF file is set to, for instance, Adobe Acrobat, and Adobe Acrobat is stored in the /users/myname/Applications folder, then it is equally valid to encode this as either "/users/myname/Applications/Adobe Acrobat.app" or simply "Adobe Acrobat". (When opening a file, Scrivener will only use the application name.)

## CustomMetaData Element

**Description:** The **CustomMetaData** element encodes any custom meta-data values associated with the binder document. Scrivener 2.0 allows the user to define custom meta-data fields to appear in the inspector and outliner columns. Information about these custom meta-data fields are encoded in the separate **CustomMetaDataSettings** subelement of the **ScrivenerProject** element of the SCRIVX file. The **CustomMetaData** subelement of **BinderItems**'s **MetaData** element encodes the document's values for the fields defined by the **CustomMetaDataSettings** element of **ScrivenerProject**. For clarification, see the example below.

**Subelements:** **MetaDataItem**.

**Attributes:** None.

**Number:** 1 if 0. If not present, the current binder document has no custom meta-data associated with it.

**Errors:** None.

**Value:** None.

**Example:**

```
<ScrivenerProject>
    <Binder>
        <BinderItem>
            <Title>My Document</Title>
            <MetaData>
                <CustomMetaData>
```

```
                    <MetaDataItem>
                        <FieldID>LOCATIONNOTES</FieldName>
                        <Value>A ruined castle.</Value>
                    </MetaDataItem>
                </CustomMetaData>
            </MetaData>
        </BinderItem>
    </Binder>
    <CustomMetaDataSettings>
        <MetaDataField ID="LOCATIONNOTES" Wraps="Yes">Location Notes</MetaDataField>
    </CustomMetaDataSettings>
</ScrivenerProject>
```

**Notes on Example:** In the above XML (which, note, would be invalid in reality because there is no Draft, Research or Trash folder, but is just provided as a simplified example), the **CustomMetaDataSettings** element at the bottom defines which custom meta-data fields have been created by the user for this project. In this case, a single custom meta-data field entitled "Location Notes" has been created by the user, and Scrivener has assigned it an internal ID of "LOCATIONNOTES". (This ID is used internally to associate it with table columns and so on, and will persist even if the user changes the name of the meta-data field - it is never seen by the user.) The **CustomMetaData** element inside the **MetaData** of the **BinderItem** element specifies that the document entitled "My Document" has had the text "A ruined castle." entered into the custom "Location Notes" meta-data field (identified by its FieldID of "LOCATIONNOTES"). i.e., the **CustomMetaDataSettings** element inside **ScrivenerProject** defines the names of custom columns in the outliner, while the **CustomMetaData** element in each **BinderItem** determines the values that will appear in such custom columns. So in the example above, for this document the user would see the text "A ruined castle." in the column labelled "Location Notes".

## MetaDataItem Element

**Description:** The **MetaDataItem** encodes the value associated with any custom meta-data fields for the binder document.
**Subelements:** **FieldID, Value**.
**Attributes:** None.
**Number:** 1 or more - there should be as many **MetaDataItem** elements inside the **CustomMetaData** element as there are values associated with custom meta-data fields for the current document.
**Errors:** None.
**Value:** None.

## FieldID Element

**Description:** The **FieldID** encodes the unique ID of a custom meta-data field defined for the project.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** One of the **ID** values defined in the **CustomMetaDataSettings** subelement of the **ScrivenerProject** element in the SCRIVX file.
**Example:** Consider the following custom meta-data settings as defined in the .scrivx file:

```
<ScrivenerProject>
    …
    <CustomMetaDataSettings>
        <MetaDataField ID="TIMEOFSCENE" Wraps="No" Color="1.0 0.0 0.0">Time of Scene</
        MetaDataField>
        <MetaDataField ID="LOCATIONNOTES" Wraps="Yes">Location Notes</MetaDataField>
    </CustomMetaDataSettings>
    …
</ScrivenerProject>
```

In this case, the only valid values for the **FieldID** element of **BinderItem MetaDataItem** elements is "TIMEOFSCENE" or "LOCATIONNOTES", as these are the only field IDs that have been defined for this project.

## Value Element

**Description:** The **Value** element encodes the value of the current custom meta-data field associated with the binder document.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** A string representing the current value of the custom meta-data field associated with the field ID specified in the **FieldID** element.

## TextSettings Element

**Description:** The **TextSettings** element encodes the text settings of the binder document. This element is only applicable to text and folder

| | |
|---|---|
| | documents (as they are the only document types to have a main text associated with them). |
| **Subelements:** | **TextMode, TextSelection, TypingAttributesRTFData, Target.** |
| **Attributes:** | None. |
| **Number:** | 1 or 0. |
| **Errors:** | None. |
| **Value:** | None. |

## TextMode Element

| | |
|---|---|
| **Description:** | The **TextMode** element encodes whether the text document encoded by the current **BinderItem** element is currently set to use scriptwriting mode or general text mode. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0. If not present, general text mode is assumed. |
| **Errors:** | None. |
| **Value:** | “Script” or “General”. |

## TextSelection Element

| | |
|---|---|
| **Description:** | The **TextSelection** element encodes the range of currently selected text for the text document encoded by the current **BinderItem** element. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0. If not present, a range of 0,0 is assumed (i.e. no selection with the cursor at the start of the text). |
| **Errors:** | None. |
| **Value:** | Two integers separated by a comma (with no space). The first number indicates the index of the character at which the selection starts, and the second number describes the number of characters across which the selection applies. So, for instance, if the word “Science” is currently selected and the “S” of “Science” is the tenth letter in the text, then the range would be 9,7 (because the tenth letter is at index 9 and the word “Science” contains 7 characters). Note that if there is no selection, the second integer should be 0. For instance, a value of 1,0 indicates that the cursor is currently inserted before the second character. |

## TypingAttributesRTFData Element

**Description:** The **TypingAttributesRTFData** element encodes the typing attributes for the text document encoded by the current **BinderItem** element. In Scrivener 2.0, a user can change the font and formatting of a blank (empty) document, and then choose to "remember typing attributes" so that when he or she returns to that blank document, the typing attributes he or she set will be retained rather than return to the defaults (as determined by the preferences). This is useful, for instance, for setting up blank template documents that may require different formatting. The **TypingAttributesRTFData** element saves the format as RTF encoded in hexadecimal.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0. If not present, the default typing attributes are used.

**Errors:** None.

**Value:** A string of hexadecimal characters. To encode the typing format, Scrivener creates an RTF file in memory with some placeholder text to which the typing attributes have been applied, then encodes this RTF data as hexadecimal. This hexademical data is then used as the value of this element. When Scrivener reads out the typing attributes, it converts the hexadecimal to RTF, then reads the text formatting from the start of the RTF data and uses this as the typing attributes.

## Target Element

**Description:** The **Target** element encodes the target settings for the text document encoded by the current **BinderItem** element - for example, the word count target.

**Subelements:** None.

**Attributes:** **Type, Notify**.

**Number:** 1 or 0. If not present, no targets have been set for the current text document.

**Errors:** None.

**Value:** An integer specifying the target for the current document in words or characters (as determined by the **Type** attribute). For instance, <Target Type="Words">5000</Target> sets a target of 5,000 words for the current document, whereas <Target Type="Characters">5000</Target> sets a target of 5,000 characters.

### Type attribute

**Description:** The **Type** attribute of the **Target** element encodes the target type

- i.e. Whether the target should be measured in words or characters.

**Required:** Yes. If no **Type** is provided, the target type is assumed to be "Words".

**Value:** "Words" or "Characters".

### Notify attribute

**Description:** The **Notify** attribute of the **Target** element encodes whether Scrivener should show some sort of notification when the user meets the specified target. Scrivener for the Mac uses the Growl framework to show a pop-up window or play a user-defined sound if **Notify** is set to "Yes".

**Required:** No. If no **Notify** attribute is provided, "No" is assumed.

**Value:** "Yes" or "No".

### AllowOver attribute

**Description:** The **AllowOver** attribute of the **Target** element encodes whether target overruns should be shown in the interface.

**Required:** No. If no **AllowOver** attribute is provided, "No" is assumed.

**Value:** "Yes" or "No".

### Buffer attribute

**Description:** The **Buffer** attribute of the **Target** element encodes how much over the target the user can go before Scrivener starts showing the overrun indicator.

**Required:** No. If no **Buffer** attribute is provided, a 0 buffer is assumed.

**Value:** An integer representing the target overrun buffer.

## MediaSettings Element

**Description:** The **MediaSettings** element encodes specific settings for the media document (e.g. PDF, image, sound and movie files) encoded by the current **BinderItem** element. It does not apply to (and therefore should not be present for) text documents.

**Subelements:** **ImageScaleFactor, ImageRotation, ImageScalesProportionally, CurrentPDFPage, OriginalURL, PlaybackPosition, IsAlias, IsSoundFile.**

**Attributes:** None.

**Number:** 1 or 0.

**Errors:** None.

**Value:** None.

## ImageScaleFactor Element

**Description:** The **ImageScaleFactor** element is only applicable to image documents. It encodes the current scale (zoom) factor of the image when displayed in one of Scrivener's editors.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** A floating point number representing the current scale factor, where 1.0 represents the original scale factor, 0.5 would scale it to half size, and 2.0 would be double size.

## ImageRotation Element

**Description:** The **ImageRotation** element is only applicable to image documents. It encodes the rotation of the image when displayed in one of Scrivener's editors.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** One of the following values: "None" for no rotation; "90CCW" for a rotation of 90 degrees counter-clockwise; "90CW" for a rotation of 90 degrees clockwise; "180" for a rotation of 180 degrees.

## ImageScalesProportionally Element

**Description:** The **ImageScalesProportionally** element is only applicable to image documents. It encodes whether or not the image should be scaled to fit the current editor rather than using the scale set by **ImageScaleFactor**. If set to "Yes", the image will expand to fit the size of the current editor (maintaining its proportions) regardless of the current scale setting (the scale slider will be greyed out in the interface).
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** "Yes" or "No".

## CurrentPDFPage Element

**Description:** The **CurrentPDFPage** element is only applicable to PDF documents. It encodes the number of the page the user was last viewing in the PDF document so that Scrivener can reopen the PDF file at the same page.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** An integer representing the last-viewed page number of the PDF file.

## OriginalURL Element

**Description:** The **OriginalURL** element is only applicable to web documents. It encodes the original URL of the web page. It is therefore only present for web pages that have been imported using Scrivener's "Import Web Page" feature.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** A string representing a web page URL.

## PlaybackPosition Element

**Description:** The **PlaybackPosition** element is only applicable to video and audio media documents. It encodes the current playback position of the media file.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** A current playback time, in seconds.

## IsAlias Element

**Description:** The **IsAlias** element encodes whether or not the file is a Mac OS X alias (pointing to another file on disk).
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** "Yes" or "No".

## IsSoundFile Element

**Description:** The **IsSoundFile** element encodes whether or not a media file contains only sound, with no video (this setting only applies to binder items with **Type** set to "Media"). This is usually not necessary, as Scrivener determines whether documents of type "Media" are sound or movie files by checking their file extensions or UTIs. However, Scrivener for Mac's audio note recording feature records the notes as .mov files, and so uses the **IsSoundFile** setting to note that these .mov files should be displayed with an audio icon rather than a movie icon. (This setting merely affects which icon is used to represent the file, and does nothing else.)

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0.

**Errors:** None.

**Value:** "Yes" or "No".

## CorkboardAndOutliner Element

**Description:** The **CorkboardAndOutliner** element encodes any corkboard and outliner settings specific to the binder document encoded by the current **BinderItem** element. For instance, it saves whether the corkboard is in freeform mode or not, the positions of any cards on the freeform corkboard, and which child documents are expanded or collapsed in the outliner.

**Subelements:** **SelectedSubdocumentIDs, OutlinerExpandedState, CorkboardSettings.**

**Attributes:** None.

**Number:** 1 or 0.

**Errors:** None.

**Value:** None.

## SelectedSubdocumentIDs Element

**Description:** The **SelectedSubdocumentIDs** element encodes the internal IDs of any descendants (contained in the **Children** element) of the current binder document that are currently selected in either the corkboard or outliner.

**Attributes:** None.

**Number:** 1 or 0.

**Errors:** None.

**Value:** A list of all IDs (as integers) of selected subdocuments separated by commas or a hyphen to indicate a continuous range. For instance, a value of "5, 12, 35-39, 46" indicates that the binder documents with the internal IDs of 5, 12, 35, 36, 37, 38, 39 and 46 should be selected when the current binder document is displayed in corkboard or outliner mode (assuming that the documents with these internal IDs are subdocuments or descendants of the current binder document).

## OutlinerExpandedState Element

**Description:** The **OutlinerExpandedState** element encodes a list of the internal IDs of all of the subdocuments of the binder document which should currently be expanded when displayed in outliner mode. Scrivener uses this information when the user selects this binder document and switches to outliner mode. It checks this list and expands any documents in the outliner which have an internal ID contained in it.
**Subelements:** **ItemID**.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** None.

## ItemID Element

**Description:** The **ItemID** element encodes the internal ID of a binder document (as defined by the **ID** attribute of **BinderItem**). It is used in the **OutlinerExpandedState** element to determine which binder documents should have their subdocuments expanded and visible when displayed in the outliner.
**Subelements:** None.
**Attributes:** None.
**Number:** There should be as many **ItemID** elements in the **OutlinerExpandedState** element as there are descendants of the current binder document that should be expanded when it is displayed in outliner mode.
**Errors:** None.
**Value:** An integer representing the internal ID of a binder document that should be expanded when viewed in the outliner. Note that whereas **SelectedSubdocumentIDs** are always stored in numerical order, **OutlinerExpandedState** stores the IDs in the order in which they appear in the outliner so that Scrivener can go through expanding them sequence - thus the IDs are

encoded differently.

## CorkboardSettings Element

**Description:** The **CorkboardSettings** element encodes information about the placement of the subdocuments of the binder document encoded by the current **BinderItem** element on the freeform corkboard.
**Subelements:** **FreeformMode, FreeformCorkboardDocuments.**
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** None.

## FreeformMode Element

**Description:** The **FreeformMode** element encodes whether or not the corkboard should be switched to freeform mode when showing the subdocuments of the binder document encoded by the current **BinderItem** element.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, assume “No”.
**Errors:** None.
**Value:** “Yes” or “No”.

## FreeformCorkboardDocuments Element

**Description:** The **FreeformCorkboardDocuments** element encodes the positions on the freeform corkboard of all subdocuments of the current binder document.
**Subelements:** **IndexCardBinderID**.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** None.

## IndexCardBinderID Element

**Description:** The **IndexCardBinderID** element encodes the internal ID, position and time stamp of a single subdocument of the binder document encoded by the current **BinderItem** element.

**Subelements:** None.
**Attributes:** **Position**, **TimeStamp**.
**Number:** If freeform mode has been used for the current binder document, there should be as many **IndexCardBinderID** elements as there are immediate subdocuments of the current binder document. If freeform mode has not been used so that there are no saved positions, or if any **IndexCardBinderID** elements are missing, Scrivener should choose a default position for affected subdocuments.
**Errors:** None.
**Value:** None.

## Position attribute

**Description:** The **Position** attribute encodes the position of a subdocument on the freeform corkboard.
**Required:** Yes. If no **Position** is provided, Scrivener should choose a default position.
**Value:** Two floating point numbers separated by a comma (with no space), representing the position of the card relative to the current fixed card width. Scrivener calculates this value by taking the upper left corner of the card and dividing its position by the current fixed card width. For instance, if a card's upper-left corner has a co-ordinate of 30,15 and the current card width is set to 60, then the saved co-ordinate will be 0.5,0.25 (30/60 = 0.5, 15/60 = 0.25). This method is used so that the card width can be used to scale up or down the freeform corkboard. So in the example case, if the user scales up the card width to 120, when positioning the card, Scrivener will place the card at position.x x 120 and position.y x 120, which will result in a position of 0.5 x 120 and 0.25 x 120 = 60,30 - which is the correct position as by scaling up the card from 60 points across to 120 points across, the user has scaled up the corkboard by a factor of 2.

## TimeStamp attribute

**Description:** The **TimeStamp** attribute encodes the time stamp of a subdocument on the freeform corkboard. The time stamp is used to determine the drawing order of cards on the freeform corkboard. Whenever a user clicks on a card to select it, or moves a card, the time stamp is updated to the current time. Cards with a more recent time stamp are drawn last. This means that if cards overlap, the most recently selected, clicked or moved card will be drawn on top.
**Required:** Yes. If no **TimeStamp** is provided, assume now.
**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm

on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

## Keywords Element

| | |
|---|---|
| **Description:** | The **Keywords** element encodes a list of all keyword IDs associated with the binder document. (Note that the keyword IDs available in the project are defined in the separate **Keywords** subelement of **ScrivenerProject** element in the SCRIVX file.) |
| **Subelements:** | **KeywordID**. |
| **Attributes:** | None. |
| **Number:** | 1 or 0 if there are no keywords associated with the current document. |
| **Errors:** | None. |
| **Value:** | None. |

## KeywordID Element

| | |
|---|---|
| **Description:** | The **KeywordID** element encodes the ID of a project keyword (these IDs are defined by the **Keywords** subelement of the **ScrivenerProject** element in the SCRIVX file). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | There should be as many **KeywordID** elements as there are keywords associated with the current document. |
| **Errors:** | None. |
| **Value:** | An integer representing a project keyword ID. |

## References Element

| | |
|---|---|
| **Description:** | The **References** element encodes any references associated with the binder document encoded by the current **BinderItem** element. References are accessed via the inspector and may contain links to other documents in the project or to external files. |
| **Subelements:** | **Reference**. |
| **Attributes:** | None. |
| **Number:** | 1 or 0 if there are no references associated with the current binder document. |
| **Errors:** | None. |
| **Value:** | None. |

## Reference Element

**Description:** The **Reference** element encodes a single reference that is associated with the current binder document.
**Subelements:** None.
**Attributes:** **BinderID**, **Destination**. (Note that **BinderID** is used for internal links - links to other documents within the project - and **Destination** is used for external links - web page URLs or external files - and so are mutally exclusive.)
**Number:** There should be as many **Reference** elements as there are references associated with the current document.
**Errors:** None.
**Value:** A string containing the title of the current reference (the user can give references any title).

### BinderID attribute

**Description:** The **BinderID** attribute of the **Reference** element encodes the internal ID of another binder document within the current project.
**Required:** *Either* the **BinderID** *or* the **Destination** attribute must be present. If neither is present, the **Reference** element refers to nothing and so is ignored.
**Value:** An integer representing the internal ID of another document in the current project, as defined by the **ID** attribute of another **BinderItem**.

### Destination attribute

**Description:** The **Destination** attribute of the **Reference** element encodes the URL to an external file or web page.
**Required:** *Either* the **BinderID** *or* the **Destination** attribute must be present. If neither is present, the **Reference** element refers to nothing and so is ignored.
**Value:** A string representing the URL to an external file or web page.

## DefaultChildTemplateID Element

**Description:** The **DefaultChildTemplateID** element encodes the ID of a document within the project templates folder that should be used as the basis for any new documents created inside this folder.
**Subelements:** **Reference**.
**Attributes:** None.
**Number:** 1 or 0 if the default should be −1 (regular text documents).
**Errors:** None.

**Value:** An integer representing the internal ID of a document inside the project templates folder, or −1 if the folder should use the default subdocument type (text files).

## UserData Element

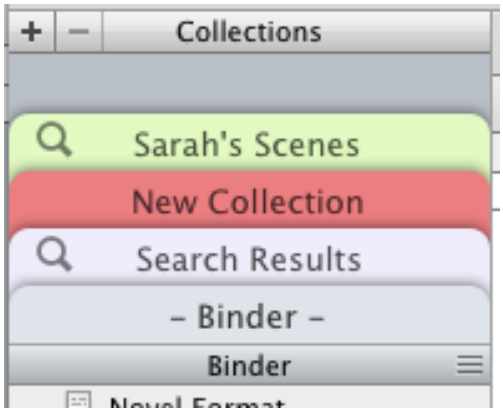
**Description:** The **UserData** element encodes and preserves data added by third-party applications (or added directly by the user). See p. 138 for details.
**Subelements:** Any custom elements added by users or third-party applications.
**Attributes:** None.
**Number:** 1 or 0 if no custom data has been added.
**Errors:** None.
**Value:** None.

## Children Element

**Description:** The **Children** element subelement of the **BinderItem** element all of the child documents of the current binder document. The **Children** element contains other **BinderItem** elements which may themselves contain **Children** elements with even more **BinderItem** subelements - in this way, the hierarchical structure of the binder is represented by the various **BinderItem** XML elements.
**Subelements:** **BinderItem**.
**Attributes:** None.
**Number:** Exactly 1 or 0 if the current binder document has no subdocuments.
**Errors:** None.
**Value:** None.

## *4.2. The Collection Element*

The **Collections** subelement of the **ScrivenerProject** element encodes information about the collections in the current project. Collections are used in Scrivener 2.0 to store saved searches and arbitrary groupings of documents from the binder. Collections can be compiled via the File > Compile feature just as the Draft can be. Collections appear as tabs above the binder; if one is selected, the content of that collection replaces the binder, so that in place of the binder there is a list of the documents contained in that collection.




Note that anything contained in a collection is also contained in the binder - collections are just a way of viewing an arbitrary selection of binder documents in isolation, either by running a search for all documents matching a certain criteria (“saved search” collections) or by assigning documents to a collection manually (“arbitrary” collections).

Note that there are four types of collections: The binder collection, which is just a tab in the collections interface used for returning to the binder; the recent search (or search results) collection, which shows the results of the last search the user ran; saved search collections, which are much the same as the recent search collection except that the user can name them and they can be used to remember any search (so that, for instance, the user can save a search of all documents from a particular character’s point of view in order to keep track of that character’s storyline); and arbitrary collections, to which the user can manually assign documents from the binder.

The binder collection and the recent search collections should *always* be available, so if the **Collections** XML element does not contain them, Scrivener will automatically add them anyway.

The **Collection** element thus contains a number of **Collection** subelements, the elements and attributes of which are as follows:

### Collection Element

**Description:** The **Collection** element encodes a single collection.
**Subelements:** **Title, SearchSettings, BinderIDs, CorkboardSettings**

**Attributes:** **Type, ID, Color**
**Number:** There should be as many **Collection** elements as there are collections in the project. Generally there should be at least two - the binder collection and the recent search collection. However, if these aren't present, Scrivener will add them automatically.
**Errors:** None.
**Value:** None.

## Type attribute

**Description:** The **Type** attribute of the **Collection** element encodes the type of collection encoded by the **Collection** element.
**Required:** Yes.
**Value:** "Binder", "RecentSearch", "Arbitrary", "SavedSearch".

## ID attribute

**Description:** The **ID** attribute of the **Collection** element encodes a unique identifier used internally to represent this collection. This ID can be any string, but it must be guaranteed to be different to the IDs of other collections in the project.
**Required:** Yes.
**Value:** A unique string value. (Scrivener for the Mac generates a universally unique identifier (UUID) for this attribute, using the CoreFoundation CFUUIDCreate() and CFUUIDCreateString() functions.)

## Color attribute

**Description:** The **Color** attribute of the **Collection** element encodes the colour to be used to represent this collection. This is used as the tab colour for the collection in the tabs view above the binder, and also as the background colour of the collection when it is selected (i.e. so that if a collection's colour is red, when the collection is selected so that it is displayed where the binder is usually shown, the binder area will have a red background colour).
**Required:** Yes. Ignored for the binder and recent search collections, though, which use a default colour. If not present, Scrivener will use a random colour.
**Value:** Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green.

## Title Element

**Description:** The **Title** element encodes the title of the collection represented by the current **Collection** element.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** A string containing the title of the current collection.

## GroupMode Element

**Description:** The **GroupMode** element encodes the view mode a collection should open in (corkboard, outliner etc). This element is usually not present, as collections are usually opened in whichever view mode was last used for a group. This setting allows users to set a particular group mode for particular collections, however.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If not present, the default view mode will be used.
**Errors:** None.
**Value:** "Corkboard", "Outliner", "Scrivenings", "Document" or "Default". This element can be omitted altogether instead of writing it out with "Default", though.

## SearchSettings Element

**Description:** The **SearchSettings** element encodes the settings for collections of the types "RecentSearch" and "SavedSearch". It should not be present for collections of other types, and if it is it will be ignored. It stores the search parameters of the search so that Scrivener can re-run the search when the user selects the collection.
**Subelements:** None.
**Attributes:** **Operator, Type, Scope, CompileSetting, CaseSensitive**.
**Number:** 1 or 0. The **SearchSettings** element must be present for collections of the "SavedSearch" type, and will usually be present for the "RecentSearch" type (the only exception being when the project is new, if no search has yet been run). It should not be present for other collection types.
**Errors:** None.
**Value:** A string containing the text for which to search. E.g. If the user searched for the word "difference", the value of the

**SearchSettings** element would be “difference”.

## Operator attribute

**Description:** The **Operator** attribute of the **SearchSettings** element determines whether the search should restrict itself to finding only exact matches, or any word contained in the search text etc.

**Required:** Yes. If not present, “Exact” is assumed.

**Value:** One of the following values: “All” (search for all words contained in the search text), “Any” (search for any of the words in the search text), “WholeWord” (only return matches for whole words; e.g. a search for “ant” would *not* return “rant”), and “Exact” (only return results that match the phrase exactly - this is the default).

## Type attribute

**Description:** The **Type** attribute of the **SearchSettings** element determines which areas of the documents in the project should be searched - i.e. whether the search should look only in the text, only in the notes, in a combination (e.g. synopses and labels), or in all text associated with documents.

**Required:** No. If not present, “All” is assumed.

**Value:** One or more of the following values, separated by a comma (with no space): “All”, “Title”, “Text”, “Notes”, “Synopsis”, “Keywords”, “Label”, “Status”, “CustomMetaData”. (Note that “All” can only appear on its own.)

## Scope attribute

**Description:** The **Scope** attribute of the **SearchSettings** element determines whether the search should be limited only to certain documents. The search can look in all documents, in documents contained inside the draft folder only, or in selected documents only.

**Required:** No. If not present, “All” is assumed.

**Value:** One of the following values: “Draft”, “Selection” or “All”.

## ExcludeTrash attribute

**Description:** The **ExcludeTrash** attribute of the **SearchSettings** element determines whether documents contained in the Trash folder should be excluded from the search.

**Required:** No. If not present, “No” is assumed.

**Value:** “Yes” or “No”.

## CompileSetting attribute

**Description:** The **CompileSetting** attribute of the **SearchSettings** element determines whether the search should be limited only to

documents with the "Include in Compile" option set (or not set).

**Required:** No. If not present, "All" is assumed.

**Value:** One of the following values: "Include" (in which case only documents with "Include in Compile" selected are searched), "Exclude" (in which case only documents *without* "Include in Compile" checked are searched) or "All" (in which case all documents are searched - this is the default).

### CaseSensitive

**Description:** The **CaseSensitive** attribute of the **SearchSettings** element determines whether the search should be case-sensitive or not.

**Required:** No. If not present, "No" is assumed.

**Value:** "Yes" or "No".

## BinderIDs Element

**Description:** The **BinderIDs** element encodes a list of internal project IDs representing the binder documents that have been added to an arbitrary collection. It should only be present for collections of the "Arbitrary" **Type**, and will be ignored in other collection types.

**Subelements:** **BinderID**

**Attributes:** None.

**Number:** 1 or 0. The **BinderIDs** element must be present for collections of the "Arbitrary" type. It should not be present for other collection types.

**Errors:** None.

**Value:** None.

## BinderID Element

**Description:** The **BinderID** element encodes a single ID representing a binder document which has been added to the collection encoded by the current **Collection** element.

**Subelements:** None.

**Attributes:** None.

**Number:** There should be one **BinderID** element for each document that has been added to the current arbitrary collection.

**Errors:** None.

**Value:** An integer representing the internal ID of a binder document in the project (as encoded in the **ID** attribute of a **BinderItem** element).

## CorkboardSettings Element

**Description:** The **CorkboardSettings** element encodes information about the placement on the freeform corkboard of the documents contained in the arbitrary collection encoded by the current **Collection** element. Note that this element is only present for collections of the "Arbitrary" type - search collections cannot be viewed in freeform corkboard mode.
**Subelements:** **FreeformMode, FreeformCorkboardDocuments.**
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** None.

## FreeformMode Element

**Description:** The **FreeformMode** element encodes whether or not the corkboard should be switched to freeform mode when showing the collection of documents encoded by the current **Collection** element.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0. If not present, assume "No".
**Errors:** None.
**Value:** "Yes" or "No".

## FreeformCorkboardDocuments Element

**Description:** The **FreeformCorkboardDocuments** element encodes the positions on the freeform corkboard of all documents contained in the current collection.
**Subelements:** **IndexCardBinderID**.
**Attributes:** None.
**Number:** 1 or 0.
**Errors:** None.
**Value:** None.

## IndexCardBinderID Element

**Description:** The **IndexCardBinderID** element encodes the internal ID, position and time stamp of a single document contained in a collection encoded by the current **Collection** element.

**Subelements:** None.
**Attributes:** **Position**, **TimeStamp**.
**Number:** If freeform mode has been used for the current collection, there should be as many **IndexCardBinderID** elements as there are documents in the collection. If freeform mode has not been used so that there are no saved positions, or if any **IndexCardBinderID** elements are missing, Scrivener should choose a default position for affected documents.
**Errors:** None.
**Value:** None.

## Position attribute

**Description:** The **Position** attribute encodes the position of a document on the freeform corkboard.
**Required:** Yes. If no **Position** is provided, Scrivener should choose a default position.
**Value:** Two floating point numbers separated by a comma (with no space), representing the position of the card relative to the current fixed card width. Scrivener calculates this value by taking the upper-left corner of the card and dividing its position by the current fixed card width. For instance, if a card's upper left corner has a co-ordinate of 30,15 and the current card width is set to 60, then the saved co-ordinate will be 0.5,0.25 (30/60 = 0.5, 15/60 = 0.25). This method is used so that the card width can be used to scale up or down the freeform corkboard. So in the example case, if the user scales up the card width to 120, when positioning the card, Scrivener will place the card at position.x x 120 and position.y x 120, which will result in a position of 0.5 x 120 and 0.25 x 120 = 60,30 - which is the correct position as by scaling up the card from 60 points across to 120 points across, the user has scaled up the corkboard by a factor of 2.

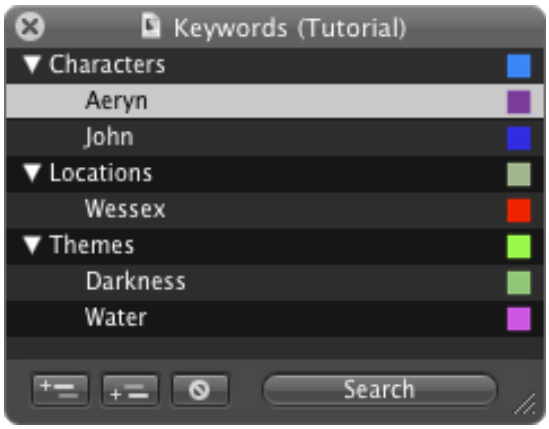
## TimeStamp attribute

**Description:** The **TimeStamp** attribute encodes the time stamp of a document on the freeform corkboard. The time stamp is used to determine the drawing order of cards on the freeform corkboard. Whenever a user clicks on a card to select it, or moves a card, the time stamp is updated to the current time. Cards with a more recent time stamp are drawn last. This means that if cards overlap, the most recently selected, clicked or moved card will be drawn on top.
**Required:** Yes. If no **TimeStamp** is provided, assume now.
**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800

indicates 8.29am on 9th December 2009 PST.

## *4.3. The Keyword Element*

The **Keyword** element encodes information about a single project keyword. Keywords can be stored in a hierarchical list in Scrivener's Keywords HUD, so one **Keyword** element may have other **Keyword** elements contained inside its **Children** subelement. (The root **Keyword** elements are stored as subelements of the **Keywords** element of the SCRIVX file.)




Elements and attributes of the **Keyword** XML element are as follows:

### Keyword Element

**Description:** The **Keyword** element encodes a single project keyword.
**Subelements:** **Title, Color, Children**
**Attributes:** **ID**
**Number:** There should be as many **Keyword** elements as there are keywords in the project. **Keyword** elements may be contained inside the **Keyword** subelement of the **ScrivenerProject** element in the SCRIVX file, or they may be contained inside the **Children** subelements of other **Keyword** elements (because the Keywords list in Scrivener is hierarchical).
**Errors:** None.
**Value:** None.

### ID attribute

**Description:** The **ID** attribute of the **Keyword** element encodes the internal project ID for the keyword. The first keyword added to the project will have an ID of 0, the next will have an ID of 1, and so on - the ID being incremented for each new keyword added to ensure they each have a unique ID.
**Required:** Yes. Without an ID attribute, a **Keyword** element is invalid.
**Value:** An integer representing the internal ID of the keyword.

## Title Element

**Description:** The **Title** element encodes the title of the keyword encoded by the current **Keyword** element.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** None.
**Value:** A string containing the title of the current keyword.

## Color Element

**Description:** The **Color** element encodes the colour associated with the current keyword. (Note that Scrivener 1.x did not associate colours with keywords - this is a 2.0 feature.)
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If not present, a random colour will be assigned.
**Errors:** None.
**Value:** Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green.

## Children Element

**Description:** The **Children** element encodes all of the child keywords of the keyword encoded by the current **Keyword** element. The **Children** element contains other **Keyword** elements which may themselves contain **Children** elements of even more **Keyword** elements, creating a hierarchical structure.
**Subelements:** **Keyword**.
**Attributes:** None.
**Number:** 1 or 0 if the current keyword has no child keywords.
**Errors:** None.
**Value:** None.

# *5. The Search Indexes XML File*

Scrivener maintains an internal dictionary (or table) containing all of the text in the project in plain text representations, so that it can search the contents of documents quickly without having to load each RTF or PDF document from disk. The dictionary (which essentially acts as a look-up table) stores a list of texts against the internal ID of each document (as encoded in the **ID** attribute of its associated **BinderItem** in the SCRIVX file), so it looks like this:

| **Key** | **Content** | **Key** | **Content** |
|---|---|---|---|
| Binder ID Number (e.g. 145) | Table of texts containing → | Title<br>Synopsis<br>Text*<br>Comments*<br>Notes<br><br>*Blank strings for non-text documents. | Title of binder document<br>Contents of synopsis<br>Textual content<br>Comments on text<br>Notes content |

The search indexes table is stored as XML in a file entitled **search.indexes** in the **[Project Name].scriv/Files** directory. It is loaded into memory when the project is opened and saved again only when the project is closed. (It is not saved during regular saves or auto-saves for speed purposes; as the content of the search indexes essentially duplicates the textual content of the project, it can easily be rebuilt again if Scrivener crashes before it is saved.)

Elements of the search.indexes XML file are as follows:

## SearchIndexes Element

**Description:** This is the root element of the search.indexes file.
**Subelements:** **Documents**
**Attributes:** **Version**
**Number:** Exactly 1.
**Errors:** A search.indexes file without a **SearchIndexes** element is an invalid document.
**Value:** None.

### Version attribute

**Description:** The **Version** attribute of the **SearchIndexes** element is currently unused but may be used in the future if the format of the search.indexes file changes.
**Required:** No.
**Value:** “1.0”.

## Documents Element

**Description:** The **Documents** element encloses multiple **Document** elements, which contain the actual search strings. The existence of this element is for future compatibility, in case elements other than document elements are added to the search indexes (there may be a project notes element in future versions, for instance).
**Subelements:** **Document**
**Attributes:** None.
**Number:** Exactly 1.
**Value:** None.

## Document Element

**Description:** The **Document** element encodes all of the searchable text for a single binder document within the project.
**Subelements**: **Title**, **Synopsis**, **Text**, **Comments**, **Notes**.
**Attributes:** **ID**.
**Number:** There should be as many **Document** elements as there are documents in the project that contain searchable text.
**Value:** None.

### ID attribute

**Description:** The **ID** attribute of the **Document** element stores the internal binder ID for the document whose searchable text is encoded by this element (this should match the **ID** attribute of a BinderItem stored in the main SCRIVX file).
**Required:** Yes. If not present, the data encoded by this element is useless because it cannot be used to look up the document associated with its text, and so should be ignored.
**Value:** An integer representing the internal ID of the document.

## Title Element

**Description:** The **Title** element encodes the title of the binder document represented by the parent **Document** element.
**Subelements**: None.
**Attributes:** None.
**Number:** 1. There may be 0 if the title is blank, but it is equally valid to have a blank **Title** element in this circumstance.
**Value:** A string containing the title of the current document.

## Synopsis Element

**Description:** The **Synopsis** element encodes the synopsis text of the binder document represented by the parent **Document** element.
**Subelements**: None.
**Attributes:** None.
**Number:** 1. There may be 0 if the synopsis is empty, but it is equally valid to have a blank **Synopsis** element in this circumstance.
**Value:** A string containing the synopsis of the current document.

## Text Element

**Description:** The **Text** element encodes a plain text representation of the textual content of the binder document represented by the parent **Document** element. Text documents, PDF documents and web documents should all have their textual content encoded as plain text in this element to enable fast searching.
**Subelements**: None.
**Attributes:** None.
**Number:** 1. There may be 0 if there is no textual content or if the document is not a text, PDF or web document, but it is equally valid to have a blank **Text** element in this circumstance.
**Value:** A string containing a plain text representation of the text of the current document.

## Comments Element

**Description:** The **Comments** element encodes a plain text representation of all of the inspector comments and footnotes associated with the main text of the binder document represented by the parent **Document** element.
**Subelements**: None.
**Attributes:** None.
**Number:** 1. There may be 0 if there are no inspector comments or footnotes associated with the current document (or if the current document is not a text document), but it is equally valid to have a blank **Comments** element in this circumstance.
**Value:** A string containing a plain text representation of all of the inspector comments and footnotes associated with the text of the current document. The text of all the comments and footnotes should be concatenated and separated using a space or a newline.

## Notes Element

| | |
|---|---|
| **Description:** | The **Notes** element encodes a plain text representation of the notes text for the binder document represented by the parent **Document** element. |
| **Subelements**: | None. |
| **Attributes:** | None. |
| **Number:** | 1. There may be 0 if there is no textual content in the notes for this document, but it is equally valid to have a blank **Notes** element in this circumstance. |
| **Value:** | A string containing a plain text representation of the notes for the current document. |

# *6. The user.lock File*

When Scrivener opens a project, it writes a **user.lock** file to the **[Project Name].scriv/Files** directory. The user.lock file contains information about the current user, and is used by Scrivener to determine whether a project is already open and in use by another user (which could cause problems) or wasn't closed properly the last time it was used (for instance, because of a program crash).

The user.lock file is deleted again when a project is closed, after all files have been successfully saved.

The user.lock file is also removed in the following circumstances:

• If the user does a Save As, the user.lock file is removed from the original project, but saved into the project's new location.
• If the user makes a backup, the user.lock file must not be included in the backed-up copy.
• If the user creates a template from the current project, the user.lock file must not be included in the template.

The user.lock file on the Mac contains the following information:

• **platform=[...]** - The current platform, i.e. "mac" or "win".
• **host=[...]** - The current computer name, e.g. "keith-blounts-macbook".
• **user=[...]** - The name of the current user.
• **uuid=[...]** - An internal UUID stored by Scrivener in the user's account on the hard disk.
• **app=[...]** - The name of the application that has the file open (e.g. "Scrivener" or "Scrivener for Windows").
• **project_path=[...]** - The full file path of the project .scriv file.
• **app_path=[...]** - The full file path of Scrivener itself.

This information is used by Scrivener on the Mac to try to determine the current status of the project and whether or not it is in use by anyone else. However, this information is arbitrary, and other implementations could include any information they want, in which case the Mac version will just assume that the project is open on another machine and warn the user as much.

The Mac version uses the information as follows:

- When Scrivener is launched, it generates an array or string containing all of the above information, ready to write it to disk. It then checks for an existing user.lock file and if one exists, loads it up and checks it against the information it had prepared.
- If there is no user.lock file present, then the project must have closed properly (except in the extreme case of the a user removing the file manually) and nobody else is using it, so the project can just be opened normally. Scrivener then writes a new user.lock file to disk so that if somebody else tries to open the file they will be alerted that it is in use.

- If there is a user.lock file but its contents are identical to what Scrivener had prepared to write to file (that is, the platform, computer, user, UUID, project path and application path stored in the user.lock file match the current user, computer, paths etc), then it is safe to assume that the project is not in use by anyone else but wasn't closed properly for some reason, perhaps because of an application crash. In this case, Scrivener rebuilds the search indexes table (checking against the docs.checksum file for which documents have changed since the search.indexes file was last saved, and rebuilding the search strings of any changed documents) and then loads the project as normal. The search indexes table is the only thing of importance that won't have been saved after an application crash.
- If the value of "app" in the user.lock file is not "Scrivener", Scrivener warns the user that the project seems to be in use by another program, and uses the value of "app" to tell the user which program currently seems to have the project open.
- If the contents of the user.lock file are identical to the current user's settings *except* for the project_path, then it is fair to assume that the user copied the project while it was open. In this case, Scrivener alerts the user that this seems to be the case, and if the user goes ahead and opens the project, Scrivener rebuilds the search indexes, updating the strings of any documents that changed since the docs.checksum file was saved.
- If the contents of the user.lock file are the same as the current settings except for the application path, Scrivener warns the user that the project seems to be open by two copies of Scrivener on the same computer. It offers to make a copy of the project from which the user can work, or the user can choose to continue and open the project regardless at his or her own risk. In either case, the search indexes will be rebuilt.
- If none of the above is true, and the user.lock file exists with conflicting information, Scrivener warns the user that the project seems to be open on another machine. The user is given the choice of working from a copy of the project or of continuing regardless at his or her own risk (again, the search indexes will need rebuilding).

Note that when Scrivener displays a warning, it can use the information in the lock file. For instance, if it seems that the project is open by someone else, the warning should tell the user that the project seems to be in use by such-and-such-a-user (using the **user** information in the lock file) on such-a-such-a-machine (using the **host** information in the lock file).

## UUID

The UUID is just a unique string that Scrivener creates and which should be unique to the current user on the current machine. On the Mac, Scrivener just generates a UUID and then stores it in a userlock.id file in the Application Support folder when Scrivener first needs to create a user.lock file on the current machine. Scrivener then reads from that userlock.id file whenever it needs to generate a user.lock file for any project, and only creates a new userlock.id file if it is wiped by the user. It is just used as an extra precaution to ensure that the user.lock file is unique for the current user on the current account of the current machine. As it only needs to be a unique string, it is entirely up to the individual implementation to decide on the format of this string and where it should be stored on the computer.

## Rebuilding the Search Indexes Table

When Scrivener updates the search indexes table, it does the following:

1. It loads the existing search.indexes into a table in memory.
2. It creates a new empty search indexes table in memory.
3. It gets a list of all binder documents in the project.
4. It iterates through every binder document in the project.
5. If the checksums for the files associated with a document differ from those saved in the docs.checksum file, or if there are no strings stored in the existing search indexes table for that document (which may be the case if the document was created during the last session), Scrivener loads the text from disk and updates the search strings for the document; if the checksums matched, it copies the strings from the existing search indexes table into the new table.
6. The new table becomes the search indexes table (the above procedure ensuring that we do not keep the strings for any documents that have been deleted in the previous session).

## *6.1. Sample user.lock File*

```
platform=mac
host=keith-blounts-macbook
user=Keith Blount
uuid=EDED7A10-DCC4-4381-A41B-14E24737DF15
project_path=/Users/keithblount/Scrivener/Development/Development Notes/
ScrivenerDevelopment.scriv
app_path=/Users/keithblount/Scrivener/Development/Code/Scrivener/build/
Release/Scrivener.app
```

# *7. The docs.checksum File*

Upon closing a project, Scrivener saves the search.indexes file (which stores the text of the project as plain text so that it can be searched by Spotlight and by Scrivener without having to load up all of the underlying files) and then the docs.checksum file. The search.indexes file is saved in the **[Project Name].scriv/Files** folder, and the docs.checksum file is saved in the **[Project Name].scriv/Files/Docs** folder.

## The docs.checksum file

The docs.checksum file stores the SHA1 value for every text (.rtf or .txt) file inside the **Docs** folder. To do this, Scrivener grabs a list of all text files inside the **Docs** folder and then iterates through them, generating a SHA1 value for each.

The docs.checksum file is a plain text file encoded using UTF8 and uses the following format:

[Filename]=[SHA1 value]

E.g:

```
1291.rtf=f9f14548dfc815fa5f3729219b81d66703522f14
1291_synopsis.txt=7367137232f3a50e17914176935830000f8425074
1292.rtf=a45c68e00cacb99607cd3c5ea234f047325eb4e7
1294.rtf=172ebec7830c36c2f0ae16e37b95ab92c230816b
1297.rtf=cefa5ea861b8a59446158fd506cd17eb8d931f2f
1299.rtf=2e4731ff4db01cde9bafe4990321c3341dffbe0f
1299_synopsis.txt=79fd7029abe8c9caf620ffaf90ec17a115a5b1ca
12_synopsis.txt=2691e12114f067587995b07ffab638a9d413eb4f
1327.rtf=5bda84d4a94774de4698b1d26b2eb27ddf93e79c
1330.rtf=009dd7e4d89508f78885e128cdbe9b468e4bd86a
1469_synopsis.txt=335370aa16a780d3cfa7ecc6ec69e36916ad8d12
1471.rtf=e0cf8cf42a3d61915dc71f4d206ccb712d1f7883
1471_synopsis.txt=d81270d11d9e93481cdd47f554dd906dcfc142d9
45.rtf=6b0a698ee804d5f30cea35b5a3642ce94c0f0746
45_synopsis.txt=06b6a9e3ac22901d58fad2faea0c71ca0a8f35fe
71_synopsis.txt=46b8fa395b834b31281b635d9ad00b6ed178954a
```

## How Scrivener for the Mac uses the docs.checksum file

Scrivener for Windows and other platforms may use the docs.checksum file to test to see if a user has changed any of the underlying files "behind Scrivener's back". Scrivener for the Mac doesn't go quite this far (it has less need to as .scriv files on the Mac are seen by the user as single files), but it does use the docs.checksum file to check for changes to files since the last session if there was a problem closing the project (because Scrivener crashed, for example).

When a Scrivener project is closed, Scrivener for the Mac saves the **search.indexes** file first and then the **docs.checksum** file. After both of these files have been saved successfully and all other saving is finished, it then removes the **user.lock** file (see p.81).

When a project is opened, Scrivener checks for the **user.lock** file. If it is present, then it indicates that either the project is open somewhere else or that it wasn't closed properly. If it wasn't closed properly, then the search.indexes file wouldn't have been saved and so will be out of sync. If the user decides to go ahead and open the project, Scrivener first loads up the search indexes as saved and then goes through the docs.checksum file, checking each SHA1 value saved there against the actual SHA1 value of any text files on disk. If it finds any with a different SHA1 value, then it loads up that file and updates the search index strings with the text from that file.

In other words, Scrivener for the Mac only uses the docs.checksum file for ensuring that the internal search indexes table is always up to date. (This works because the docs.checksum file is only saved *after* the search.indexes file has been saved successfully.)

Note that Scrivener for the Mac does not save SHA1 values for non-text files, such as PDF documents or images. This is unnecessary as these are static and so their content cannot (or at least is very unlikely to) change in a way that will affect the search (even if the user edits a PDF file by adding an annotation, the annotation wouldn't be searchable anyway). For non-text files, when rebuilding the search indexes Scrivener just checks to see if it already has a saved search string - if so, it does nothing and assumes that it will not have changed. This is mainly for speed purposes, as saving the SHA1 value for many large PDF or image files can take a long time and slow down the closing of a project considerably.

# *8. The version.txt File*

The **version.txt** file is a plain text (UTF8) file inside the \Files folder that stores a number representing the internal *project* version of the current project. Scrivener uses this to determine whether the project was created or saved using an older version of Scrivener (in which case the project may need updating in order to be opened), a newer version (in which case Scrivener may have to tell the user that the project cannot be opened unless Scrivener is updated) or the same version as is being used to open the project (in which case the project can be opened without any conflicts).

It is important to note that the project version number stored in this file does not correspond to the version of Scrivener that the user sees in Scrivener's About panel. The project version number is used and defined internally, and is only incremented whenever a change is made to the file format that requires older projects to be updated. This is, in fact, its sole purpose - it provides Scrivener with a way of knowing whether or not a project should be updated (or if a file cannot be opened because it was created in a newer version).

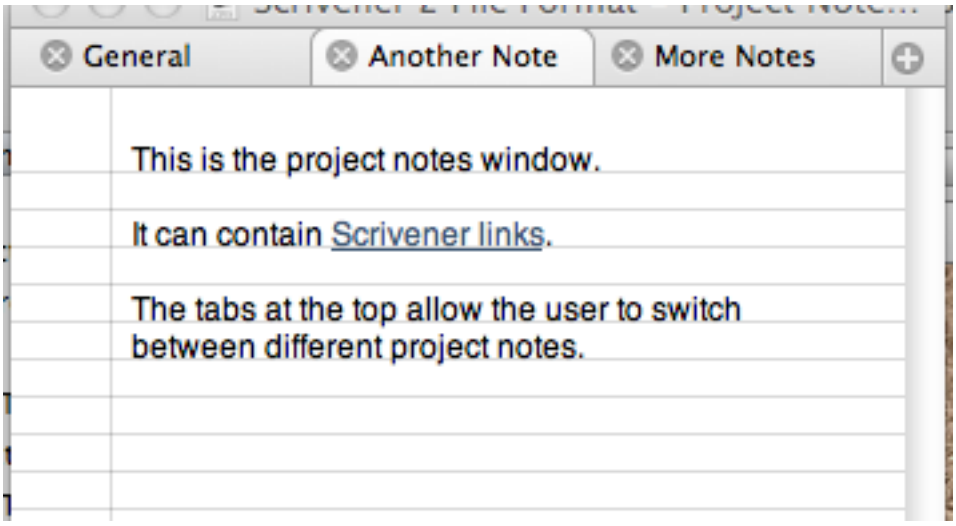
The project version numbers that have been used in release versions of Scrivener to date are as follows:

| **Scrivener Version** | **Project Version Number** |
|---|---|
| Scriver Gold | 0 |
| Scrivener 1.0 − 1.03 | 1 |
| Scrivener 1.10 − 1.5x | 4 |
| Scrivener 2.0-2.5 | 16 |
| Scrivener 2.6 onwards | 18 |

Thus Scrivener projects that were created by, or are intended to be opened in, the latest version of Scrivener should contain a **version.txt** file inside the \Files directory that contains the number **16** and nothing else.

# 9. The Project Notes XML File

The proejct notes are stored as a collectoin of RTF files referenced by an index.xml XML file. Although in Scrivener 1.x each project could only have a single set of project notes associated with it (that is, a single rich text file for storing general project notes), as of 2.0 a project can have multiple project notes. The project notes window looks like this:




(Project notes can also be accessed in the inspector via a pop-up button that allows the user to select a project note object to view in place of the document notes.)

The term "project notes" is thus now used to describe a set (array) of note objects. Each project note object stores:

- The title of the note (which appears in the tab bar of the project notes window or the pop-up list in the inspector).
- The text of the note, including Scrivener links.
- A unique internal ID number.
- The current range of text that has been selected by the user (or insertion point location).

All of this information is encoded as XML in the **index.xml** file, which is stored inside the .scriv/Files/ProjectNotes folder. For each note encoded in the index.xml file there is an associated RTF file also contained in the .scriv/Files/ProjectNotes folder. These RTF files use the following file name format:

Note-[ID].rtf

Where [ID] is replaced by the internal ID number of the note. For example, a project note with the internal ID of 12 will be saved as .scriv/Files/ProjectNotes/Note-12.rtf. Note that if there are no project notes associated with the project, the ProjectNotes folder may not exist.

Elements of the **ProjectNotes/index.xml** file are as follows:

## ProjectNotes Element

**Description:** This is the root element of the ProjectNotes/index.xml file.
**Subelements:** **Note**
**Attributes:** **Version**
**Number:** Exactly 1.
**Errors:** An index.xml file without a **ProjectNotes** element is an invalid document.
**Value:** None.

### Version attribute

**Description:** The **Version** attribute of the **ProjectNotes** element is currently unused but may be used in the future if the format of the ProjectNotes/index.xml file changes.
**Required:** No.
**Value:** "1.0".

## Note Element

**Description:** The **Note** element encodes a single project note object - the text, title, unique ID and selected range of one of the project notes.
**Subelements:** **Title, RTF, SelectedTextRange**
**Attributes:** **ID**
**Number:** There should be as many **Note** elements as there are project notes in the project.
**Value:** None.

### ID attribute

**Description:** The **ID** attribute of the **Note** element stores a unique internal ID associated with this note. The ID is an integer, and each project note should have a different ID (note that these IDs are not related to the internal IDs of binder documents). The first note should have an ID of 10, the next of 11 and so on, atlhough any number can be used so long as it is not less than 10. The ID provides a convenient way of associating a project note with a user interface element. For instance, in the inspector notes pane, different project notes can be selected from a pop-up menu. The items of the menu are each associated with a tag that corresponds to a note ID, so that when an item is selected, Scrivener is able to look up the note associated with that ID in order to load it. The same happens with tabs in the tabbed project notes window, where each tab has an ID associated with it as an internal identifier so that when clicked, Scrivener

knows which note to load. (Because these ID numbers are used as tags in parts of the interface, the numbers 0-9 are reserved for other interface item tags - thus the ID of a project note should *never* be less than 10.) Note that the ID is also used in the file name of the RTF file containing the actual text of a note, and this is used by Scrivener to locate the text on disk. Thus, a note with an ID of 24 is saved in the ProjectNotes folder using the file name Note-24.rtf.

**Required:** Yes.

**Value:** An integer which represents the unique internal ID of the current project note (i.e. it must be different to the ID of other project **Note** elements). Note that for reasons pertaining to the Mac version interface, the lowest ID should be 10 - there should be no project notes that have an ID number of lower than 10.

## Title Element

**Description:** The **Title** element encodes the title of the project note that appears in the inspector pop-up menu and in the tabs of the project notes window.

**Subelements:** None.

**Attributes:** None.

**Number:** Exactly 1.

**Value:** A string representing the note's title.

## SelectedTextRange Element

**Description:** The **SelectedTextRange** element stores the range of the text that has been selected by the user - that is, the range of characters the user has highlighted for editing. If the user has made no selection, the length of this range will be 0 and instead represents the position of the insertion point. All ranges have a zero-based index, so the fourth character is at index 3, for instance.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0. If zero, you may assume the insertion point is either at the beginning or end of the file - this is left up to the individual implementation.

**Value:** Two integers separated by a comma (with no space). The first number indicates the index of the character at which the selection starts, and the second number describes the number of

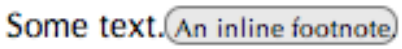
characters across which the selection exists. So, for example, if the words “Romeo and Juliet” are selected and the “R” of “Romeo” is the 21st letter in the text, then the range would be: 20,16 (because the 21st letter is at index 20 and “Romeo and Juliet” contains 16 characters). If there is no selection and the insertion point (blinking cursor) is placed before the 30th character, the value would be 29,0 (because the 30th character is at index 29 and there is no selection, so the length of the range is 0).

# 10. Text Document Files

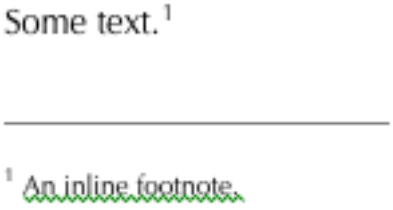
Text in Scrivener has several custom attributes. This chapter describes the custom features used in Scrivener text documents and how they are saved.

## 10.1. Inline Footnotes

Inline footnotes look like this:



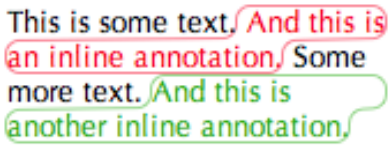

When exported, inline footnotes become real footnotes in formats that support footnotes. So if the above was exported to Word as RTF, it would look like this:



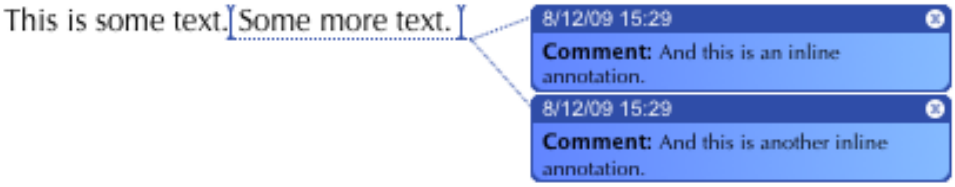

## 10.2. Inline Annotations

Inline annotations look much like inline footnotes except that they can be of any colour, the text is coloured too, and there is no fill colour.

For example:




Inline annotations are used for making notes inside the text, and are usually removed when exporting or printing. The user can choose to include annotations in the print or export, although their appearance in the final document will depend on the format. If the user exported the above to RTF and opened it in Word, it would look like this:



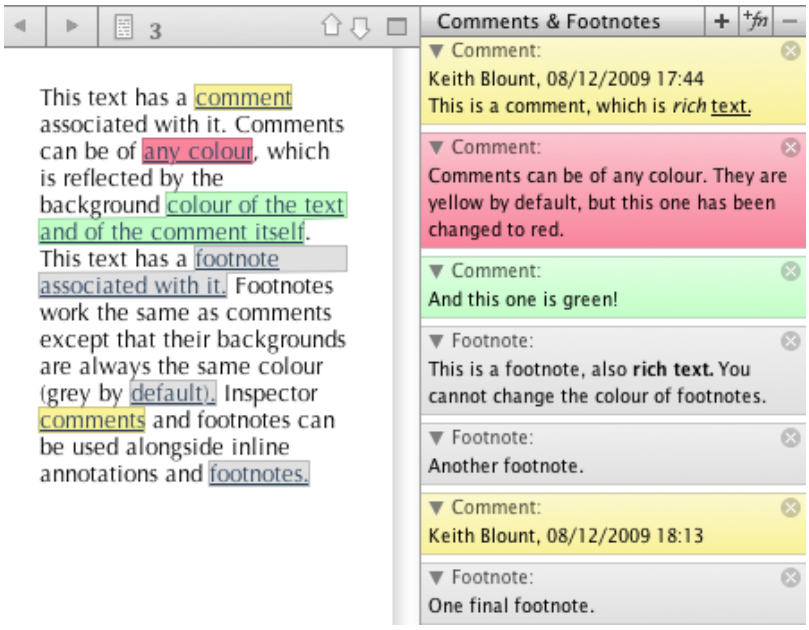
If the user exported it to RTFD or printed directly from Scrivener, it would look like this:

This is some text.[ And this is an inline annotation.]
Some more text. [And this is another inline annotation.]

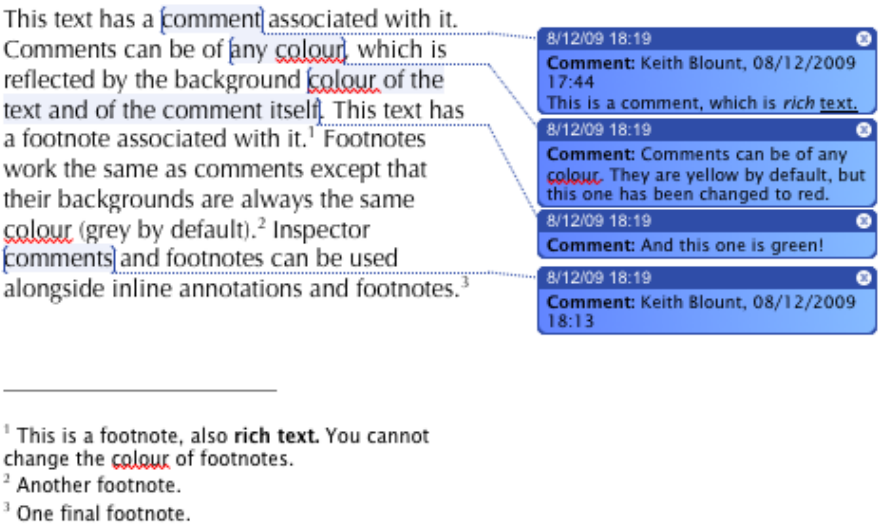
## *10.3. Inspector Comments and Footnotes*

In addition to inline footnotes and annotations, Scrivener 2.0 also supports new types of comments and footnotes that are displayed in a new pane of the inspector. These new comments and footnotes are entirely separate to inline anntotations and footnotes, although the user can choose to convert between them via the Format > Convert menu (and the user can specify in the Preferences which format - inline or inspector - should be used for imported documents). I expect that most users will use inspector comments and footnotes in the future, but inline footnotes and comments will remain for compatibility with earlier versions and those who like them - they may possibly be phased out in a future version (3.0 or above) depending on user feedback, although I am fond of the inline annotations myself so this is doubtful.

Note that text marked with an inspector comment or footnote also has a link attribute and tooltip associated with it. Hovering the mouse over the text will bring up the comment in a tooltip; clicking on the text will open the inspector and automatically jump to the associated comment.




As with inline footnotes and annotations, how inspector comments and footnotes appear in compiled documents will depend on the export format. Again, if the above were exported to Word as RTF, this is how it would appear in Word:




## *10.4. Scrivener Links*

Scrivener links are wiki-style hyperlinks that look exactly the same as web links, except that instead of storing a URL they store the internal ID number of another document within the project. Clicking on the link may open the linked document in the current editor, in the other editor pane, or inside a QuickLook HUD window, depending on the user's Preferences.

## *10.5. How Text Files Are Saved*

Scrivener stores all text files inside the /Files/Docs subfolder of the .scriv file.

The main text and notes text for documents are saved in the RTF file format:

http://msdn.microsoft.com/en-us/library/aa140277(office.10).aspx

An older but easier-to-read version can be found here:
http://www.biblioscape.com/rtf15_spec.htm

Synopses are saved as plain text .txt files using the UTF8 encoding.

All files are saved using the internal binder ID number as the basis for their file names.

### Text Files

Any *text* file may have the following files associated with it:

| File | Filename example | Description |
|---|---|---|
| Main text file | 18.rtf | An RTF file that contains the main text for text documents - the text that will appear in the main section of the editor. The RTF used in Scrivener supports bullets, tables, images and most other common rich text formatting. Certain special mark-up is used to save some text elements, such as inline annotations and footnotes, as described in “RTF Inline Mark-Up” on p. 99. |
| Notes file | 18_notes.rtf | An RTF file that contains the notes text associated with a document - the text that appears in the notes pane of the inspector. Again, this file supports images, tables etc. |
| Synopsis file | 18_synopsis.txt | A plain text file using UTF8 encoding that stores the text that appears in the synopsis associated with the document (which appears in the index card in the inspector and on the corkboard, and also in the outliner synopsis column). |
| Image card file | 18_icImage.jpg | As of Scrivener 2.0, it’s not just image files that can be represented as images on the corkboard. Any file can have an image associated with it, and it can be displayed in the inspector in place of the synopsis. If an image is displayed in the inspector instead of the synopsis, then the image is used on the corkboard instead of the synopsis too. Image card files can be of any supported image file type (.jpg, .png, .tif, .bmp etc). |
| Comments file | 18.comments | The .comments file is an XML file that stores information about Scrivener inspector comments and footnotes. |

## Non-Text Files

Any *non-text* file can have exactly the same files as above, except that instead of the main text file they will have a different file type. For instance, if the item that has the internal ID 27 is a PDF file, then its file structure may consist of:

27.pdf
27_notes.rtf
27_synopsis.txt
27_icCard.png

Non-text files do not have .comments files associated with them.

(See “Non-Text Documents” on p.105 for more information on non-text documents such as image and PDF files.)

## Empty Files

To keep things as tight as possible, Scrivener only saves to disk files that have content and removes empty files. So, for instance, a text file that has nothing typed in it as yet, no notes, a blank synopsis and no image card file associated with it, would have no representation on disk in the \Files\Docs folder - there is no content so no files are saved.

Thus, a file with the internal ID 98 that is an image file with notes but no synopsis and no image card associated with it (note that image documents can still have an image card associated with them that would override the default representation on the corkboard), might have these files in the .scriv/Files/Docs folder:

98.tif
98_notes.rtf

## *10.6. Apple RTF Extensions Used in Scrivener*

Apple has added a number of its own extensions to the RTF format, as detailed on their website here:

http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/AttributedStrings/Tasks/RTFAndAttrStrings.html

Although Scrivener for the Mac uses Apple's RTF loading and saving code as the basis for storing and opening RTF files, code has been added and amended to make the RTF that Scrivener uses as standard as possible. Readers of RTF files created by Scrivener can therefore, for the most part, safely ignore Apple's extensions to the format.

That said, the following Apple RTF commands may cause some minor discrepancies when opening Scrivener RTF documents with other RTF readers, so it is important to be aware of them:

**\slleadingN**
**\slmaximumN**
**\slminimumN**
**\ulstyleN**
**\strikecN**

**\slleadingN** gets written out if the user chooses to set a specific point-spacing between lines within a paragraph using "inter-line spacing" in the OS X spacing panel rather than use a line height multiple. This is an advanced feature and it is much more common to use a line height multiple, but if this command is present but ignored by the RTF reader then it could lead to line spacing not appearing as the user expects.

**\slmaximumN** and **\slminimumN** are only written out if the user sets different values for "At least" and "At most" in the line height settings of the OS X spacing panel. If both values are empty or the same (which would usually be the case), the values get written out using the single, standard **\sl** RTF command in RTF. In the worst-case scenario of the **\slmaximumN** or **\slminimumN** commands being present but unsupported by the RTF reader, the line spacing or leading may not appear as the user expects.

**\ulstyleN**: Scrivener supports underline-by-word and double underlines, so these settings may be lost in RTF readers that do not support the **\ulstyleN** command.

**\strikecN** is written out if the user has applied a strike-through colour. Although the user could conceivably use the OS X font panel to apply a colour to a strikethrough, Scrivener's out-of-the-box support adds a strikethrough of the same colour as the text. If an RTF reader does not support this command, all that will happen is that the strike-through colour added via the OS X font panel will be lost.

**Note on soft line breaks**

The Apple RTF exporter saves out soft line breaks using the unicode character code:

`\uc0\u8232`

Most RTF exporters and importers use `\line` instead. Scrivener for the Mac can read either, but because it uses the Apple exporters as its base, it outputs soft line breaks as `\uc0\u8232`, so other versions of Scrivener should be able to intepret both `\uc0\u8232` and `\line` correctly as soft line breaks.

There may be other minor discrepancies as a result of Apple's RTF extensions, but they will be negligible and unnoticeable in all but the most extreme cases.

## *10.7. Saving RTF Custom Elements*

Although Scrivener uses RTF files to store text, it has to save certain custom text elements and does this in two different ways:

**Inline annotations and footnotes** are saved as RTF Inline Mark-Up.

**Inspector comments and footnotes** are saved in an assocatied .comments XML file. Text linked to comments is saved into the RTF with a custom URL link of the format "scrivcmt://UUID" (where the UUID matches the ID attribute of the associated Comment element). Upon loading, Scrivener parses the text looking for these custom links, and then looks up the associated comments from the .links file to apply them.

**Internal document links** are saved into the XML as URLs using the following format: scrivlnk://*UUID*, where *UUID* is the internal UUID of the document. E.g. "scrivlnk://CFB15F5A-39AC-4AD9-A78E-2DB585D5BD5D".

*Note:* Although RTF supports footnotes and comments, the Cocoa RTF reader/writer does not, which is why these elements are saved in different ways. (Scrivener for the Mac can import and export RTF documents with these elements intact, but it is not reliable enough for an internal format that must remain consistent between saving and loading.)

## *10.7.1. RTF Inline Mark-Up*

In general, Scrivener uses standard RTF to save rich text files, but a couple of things should be taken into consideration:

### Images

Images must be saved as hexadecimal PNG or JPG data inside RTF files.

E.g:

```
{\*\shppict {\pict {\*\nisusfilename lit-lat-title} \picw109 \pich52
\jpegblip ffd8ffe000104a46494600010100000...a2803fffd9}}
```

Or:

```
{\*\shppict {\pict {\*\nisusfilename Facebook} \picw39 \pich48 \pngblip
89504e470d0a1a0a000000...44ae426082}
```

They should *not* be saved as binary data or as any other format than \jpegblip or \pngblip.

Image file names can optionally be stored within the RTF using the control ***\nisusfilename**. This is not part of the standard RTF specications and is hence prefixed with ***** rather than **** to indicate that it can be ignored by many RTF parsers. The reason it is called "**nisusfilename**" is that it was originally used by Nisus Writer (www.nisus.com) for their word processor, and as there is no standard RTF control for storing image file names, it makes sense to use one already defined by another word processor.

Storing image file names is not essential - if this filename is not provided then Scrivener will name files internally (Image1, Image2 etc). However, the ***\nisusfilename** should be supported because it means that images retain meaningful names when exported to other file formats (for example, when exporting to HTML, the image files get exported as separate files, so it is better that they have meaningful names rather than Image1.jpg, Image2.png etc). (Note that the image file name itself could contain control characters such as \bullet, \{ and so on, as it will have been grabbed from the imported image file name, so this must be taken into consideration when reading the file name.)

### PDF Files

The RTF format does not support the embedding of PDF data, so for this Scrivener uses its own, custom RTF control: \scrivenerpdf. A file name is also included with the \pdffilename tag. For instance:

```
{\*\scrivenerpdf {{\*\pdffilename FileName}
255044462d312e340d25e2e3cfd30d0a312030206f626a3c3c2f4d6574616461746120392030
020522f506167657320322030205…}}
```

## Symbolic Link Images

The user may choose to enter images into the text as symbolic links (e.g. by using Edit > Insert > Image Linked to File… or by holding the Control key down while dropping an image into the text). Symbolic link images are not embedded into the file; instead, only a reference to the original image on disk is stored. The advantage of symbolic link images is that any changes the user makes to the image on disk will be reflected in the image in Scrivener's text. The disadvantage is that if the user removes the image on disk, or takes the Scrivener project to another machine, the image can no longer be displayed in the text. Because RTF support for linked images is overly complex for Scrivener's needs, Scrivener doesn't save them RTF commands. Instead, upon save, the link destination and image size are saved into the file as regular text using this format:

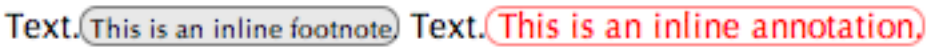
{$SCRImageLink[w:*N*;h:*N*]=*PATH*}

(Where the blue italicised text represents the width, height and path to original image respectively.) Thus, were you to open a Scrivener RTF file containing a symbolic link image in a regular RTF editor such as Word, you might see something like this:

{$SCRImageLink[w:441;h:653]=/Users/myname/Pictures/myimage.jpg}

## Inline Annotations and Footnotes

Inline annotations and footnotes look like this:




They are saved using mark-up in the rich text itself. *Before* saving to RTF, Scrivener goes through the text looking for inline annotations and footnotes and adds mark-up around them in the rich text string, then saves the marked-up rich text to RTF. Were you to open a Scrivener RTF file containing the above text from the .scriv/Files/Docs folder in a regular RTF word processor such as Word, here is what you would see:

```
Text.{\Scrv_fn= This is an inline footnote.\end_Scrv_fn} Text.{\Scrv_annot \color=
{\R=1.000000\G=0.000000\B=0.000000} \text= This is an inline annotation.
\end_Scrv_annot}
```

Inline footnotes are thus prefixed with:

```
{\Scrv_fn=
```

And suffixed with:

```
\end_Scrv_fn}
```

Inline annotations are prefixed with:

```
{\Scrv_annot\color={\R=[RedComponentAsFloat]\G=[GreenComponentAsFloat]
\B=[BlueComponentAsFloat]}\text=
```

(Where [RedComponentAsFloat], [GreenComponentAsFloat] and [BlueComponentAsFloat] are floating point numbers between 0.0 and 1.0.)

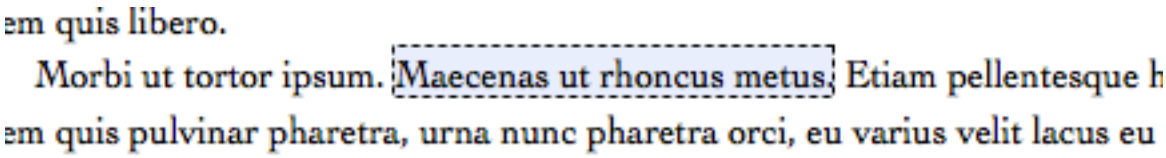
And suffixed with:

```
\end_Scrv_annot}
```

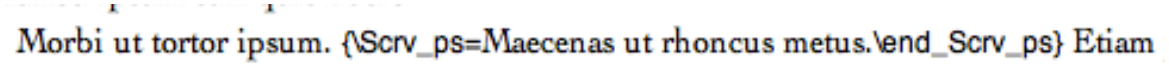
*Note*: This mark-up is in the rich text itself, *not* in the underlying RTF, despite the use of RTF-like tags. If you typed the above text into Scrivener, then closed the project and reopened it, the text would appear as a footnote or annotation and the mark-up would be gone.

## Preserve Formatting

Preserve formatting blocks in Scrivener look like this:




As with inline annotations and footnotes, these are saved using mark-up in the rich text itself. Were you to open up one of Scrivener's internal RTF files that contains a preserve formatting block in a regular rich-text editor, this is what it would look like:




## Keep-with-Next

Scrivener allows paragraphs to be be set to be kept together with the next paragraph (for instance so that character names in screenplays don't become stranded when the dialogue that follows them gets pushed onto the next page - by setting "keep with next" for the character names, the line containing the character name will get pushed down too in this instance). Although keep-with-next is supported by RTF, when saving text files inside the project package, Scrivener for the Mac does not use the RTF code but instead a custom control word. For paragraphs that should be kept-with-next, Scrivener places the control word **<$ScrKeepWithNext>** at the beginning of the paragraph. This control word is stripped out and the correct keep-with-next formatting applied when Scrivener reads RTF files from within its project package.

## *10.7.2. The .comments XML File Format*

.comments files are XML files that store the inspector comments and footnotes (which also appear in the text as links) for text documents.

> **Note**
> Text linked to comments is saved into the RTF with a custom URL link of the format "scrivcmt://UUID" (where the UUID matches the **ID** attribute of the associated **Comment** element). Upon loading, Scrivener parses the text looking for these custom links, and then looks up the associated comments from the .comments file to apply them. Any comments or footnotes in the .comments file that do not match a "scrivcmt://…" link in the text are ignored.

Note that .comments files are stored separately for each document.

.comments files contain the information about comments and footnotes. Note that these are the comments and footnotes that appear in the inspector (see p.<$p>) and which have links in the main text, as opposed to inline annotations and footnotes which have already been described (see "RTF Inline Mark-Up"on p.<$p>).

Note that snapshot XML files can also contain a **Comments** element, which encodes the inspector comments and footnotes existing in the main text of a document at the time the snapshot was taken.

Elements of the .comments XML file are as follows:

### Comments Element

| | |
|---|---|
| **Description:** | The **CommentsAndFootnotes** element stores all inspector comments and footnotes for text documents. This element should therefore *not* be present for non-text documents. |
| **Subelements:** | **Comment** |
| **Attributes:** | None. |
| **Number:** | 1 or 0 if the .comments document is associated with a non-text document or with a text document that has no inspector comments or footnotes. |
| **Value:** | None. |

### Comment Element

| | |
|---|---|
| **Description:** | The **Comment** element encodes a comment or footnote associated with the text but displayed in the inspector. |
| **Subelements:** | None. |
| **Attributes:** | **ID, Footnote, Number, Collapsed, Color** |
| **Number:** | There should be as many **Comment** elements as there are linked |

(inspector) comments and footnotes in the main text of the associated document. If there are no **Comment** elements, no .comments file should exist.

**Value:** A string representing the rich text of the comment or footnote encoded in RTF format, enclosed in a CDATA block.

## ID attribute

**Description:** Encodes the unique identifier associated with the current comment. Scrivener uses this to ID to look up the location of the comment in the text - the location is saved in the text using a custom URL link with the format scrivcmt://UUID.

**Required:** Yes.

**Value:** A unique identifier string.

## Footnote attribute

**Description:** Determines whether the comment encoded by the current **Comment** element is a footnote or not.

**Required:** No. If not present, the comment should be treated as a normal comment and *not* a footnote.

**Value:** "Yes" or "No".

## Number attribute

**Description:** If the comment encoded by the current **Comment** element was exported or printed as a footnote in the last Compile, this attribute encodes the footnote number that was assigned in the exported or printed document. The user can use this number to cross-reference comments with footnotes in the compiled document.

**Required:** No.

**Value:** An integer determining the footnote number that was used for this comment in the last Compile, or −1 to indicate that no number is assigned.

## Collapsed attribute

**Description:** Determines whether the comment or footnote encoded by the current **Comment** element should be collapsed in the inspector. Comments and footnotes in the inspector can be collapsed or expanded by clicking on a disclosure triangle. This has no effect on the content or use of the comment and describes only a temporary visual state.

**Required:** No. If not present, the comment should be considered as expanded (*not* collapsed).

**Value:** "Yes" or "No".

## Color attribute

**Description:** The **Color** attribute encodes the colour associated with a comment.

**Required:** No. If the **Color** attribute is not incluced, then the comment uses the default colour. Note that although footnotes all appear in Scrivener using the same background colour (grey by default), they may still encode a colour. This is because Scrivener can convert comments to footnotes and vice versa, so footnotes may encode the custom colour they used as a comment so that if they are converted back to being a comment they can use their former custom colour.

**Value:** Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green.

# *11. Non-Text Documents*

Scrivener stores all non-text (research) files alongside the text files inside the /Files/Docs subfolder of the .scriv file. Non-text documents are saved in exactly the same way as text files except that in place of a main text RTF file, they have a different file - such as a PDF or image file.

See “How Text Files Are Saved” on p.94 for information on how the notes, synopsis and index card image files associated with non-text documents should be saved. As specified there, *non-text* documents can have exactly the same files associated with them as text documents, except that instead of the main text file they will have a different type of file. For instance, if the item that has the internal ID 27 is a PDF file, then its file structure may consist of:

27.pdf
27_notes.rtf
27_synopsis.txt
27_icCard.png

## Recognised Research Document Types

Other than text documents, Scrivener can display the following document types in its editors:

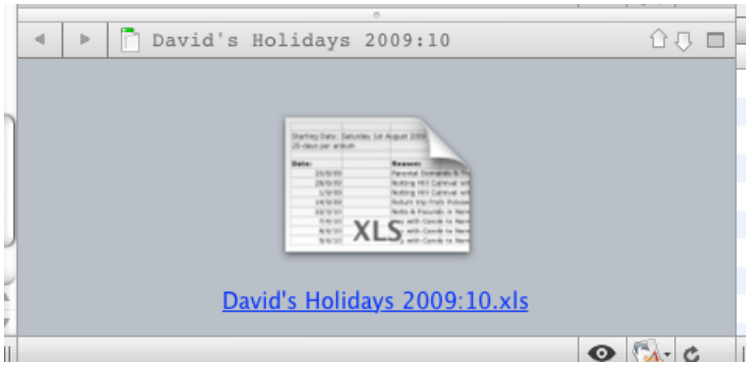
- Image files (.tif, .jpg, .bmp, .png, .gif and most other common image file types).
- PDF files.
- Sound and movie files (any file format supported by the QuickTime framework).
- Web pages.

## Note on Web Pages

Note that web pages inside Scrivener are stored in the WebKit’s .webarchive format, because this format can encode an entire page, including images, in a single file (although which elements will be included in the .webarchive file will depend on the version of the WebKit available on the user’s machine). The .webarchive format has been supported in Safari on the Mac since version 1, and on Windows since version 4. Scrivener uses the WebKit to view these files.

## Other Research Documents

As of version 2.0, Scrivener can import any file type, even those not directly supported and which cannot have their contents opened in Scrivener’s editors. Such files appear as an icon and a link in Scrivener:




If the user double-clicks on the icon, or clicks on the link, the document is opened in an external application. Scrivener for the Mac running Snow Leopard or later also allows the user to look inside these files without leaving Scrivener by using the QuickLook framework.

# *12. Snapshot XML Files*

In Scrivener, a "snapshot" is an earlier version of a text document. At any time while editing a text file, the user may "take a snapshot" of it. When the user does so, Scrivener archives a version of the text as it is at the moment the snapshot is taken so that the user may return to and restore that version of the text at any time in the future. In this way, a user can take a snapshot and then make edits confident that should he or she dislike the edits, he or she can return to the earlier version easily. Any one text document can have any number of snapshots associated with it, but all the snapshots for a single text file are stored in a single .snapshots directory.

.snapshots folders contain an index.xml XML file that stores information about the snapshots of a single text file, along with RTF files for the text of each snapshot taken. Only text files have snapshots associated with them, and only text files that have had snapshots taken of them will have an associated .snapshots folder saved in the project file.

**Each text file has a maximum of *one* .snapshots directory associated with it. Each .snapshots folder may contain multiple snapshots for its associated text document.**

Each snapshot of a text document contains the following information inside the .snapshots/ index.xml file:

- The date and time the snapshot was taken.
- The title assigned to the snapshot (optional).
- Information about any Scrivener links, comments and footnotes in the text at the time the snapshot was taken.

In addition to the index.xml file in the .snapshots directory, there is an RTF file for each snapshot taken that contains the text as it was at the time the snapshot was taken. The RTF file of a snapshot uses the date as the file name in the format:

YYYY-MM-DD-hh-mm-ss-offset.rtf

The hyphen befor the offset may become a plus sign for dates with a positive offset. For instance, a snapshot taken on the 19th July 2010 and 8.57am GMT may have the following RTF file associated with it:

2010-07-19-08-57-15-0000.rtf

A snapshot taken on 20th May 2009 at 5.28pm PST (which is eight hours behind GMT) may have the following RTF file associated with it:

2009-05-20-17-28-35-0800.rtf

And a snapshot taken on 12th January 2010 at 11.49am WST (which is eight hours ahead of GMT) may have the following RTF file associated with it:

2010-01-12-11-49-37+0800.rtf

Elements of the .snapshots/index.xml file are as follows.

## Snapshots Element

**Description:** This is the root element of a .snapshots/index.xml file.
**Subelements:** **Snapshot**
**Attributes:** **Version**
**Number:** Exactly 1.
**Errors:** A .snapshots/index.xml file without a Snapshots element is an invalid document.
**Value:** None.

### Version attribute

**Description:** The **Version** attribute of the **Snapshots** element is currently unused but may be used in the future if the format of .snapshots/index.xml files changes.
**Required:** No.
**Value:** "1.0".

## Snapshot Element

**Description:** A **Snapshot** element encodes a single snapshot of the text document with which the .snapshots/index.xml file is associated.
**Subelements:** **Title, Date, RTF, Comments**
**Attributes:** None.
**Number:** There should be as many **Snapshot** elements as there are snapshots of the document.
**Value:** None.

## Title Element

**Description:** The **Title** element stores the title of the **Snapshot** element.
**Subelements**: None.
**Attributes:** None.
**Number:** 1 or 0 if the snapshot has no title associated with it. Even untitled snapshots should generally have a title of "Untitled Snapshot", but this element is optional.
**Value:** A string containing the title of the snapshot.

## Date Element

**Description:** The **Date** element stores the date and time the snapshot was taken.

**Subelements:** None.

**Attributes:** None.

**Number:** 1. Every **Snapshot** element should have one **Date** subelement, although it is left up to the individual implementation how to deal with the lack of this element. The Mac version assigns the current date and time to a snapshot if no **Date** element is found, but the Mac version also ensures that it always encodes a **Date** element for every snapshot.

**Value:** A date stored in the format YYYY-MM-DD hh:mm:ss +offset from GMT. E.g. 2009-11-29 18:59:27 +0000 indicates 6.59pm on 29th November 2009 GMT. 2009-12-09 08:29:38 −0800 indicates 8.29am on 9th December 2009 PST.

## Comments Element

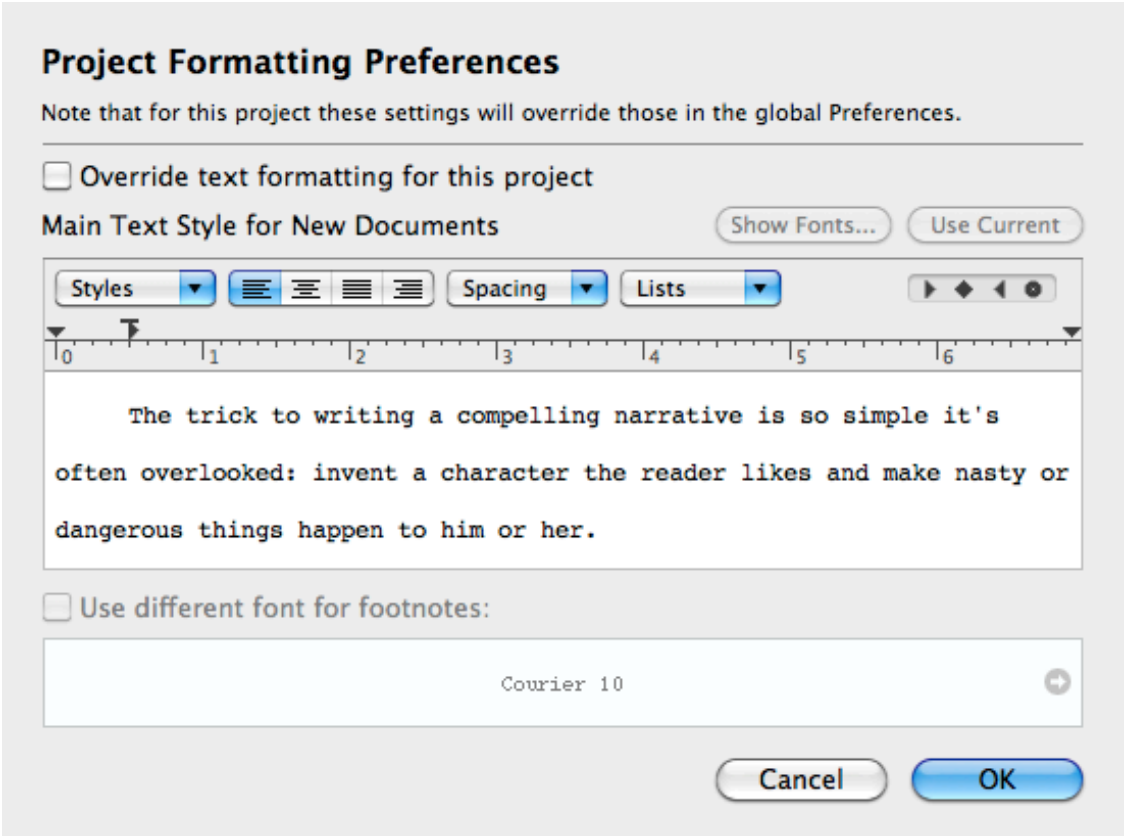
**Description:** The **Comments** element encodes all comments and footnotes for the text of the snapshot. The **Comments** element should be exactly the same as the **Comments** element in the .comments XML file for the document at the time the snapshot was taken. Refer to the file format description for .comments files on p. 102 for a description of the subelements of the **Comments** element.

**Attributes:** **Version**

**Value:** None.

# *13. The Project Preferences XML File*

Scrivener allows the user to set certain options via the Preferences (as is standard on OS X). These include preferences for the formatting to use in new documents (font, paragraph style and so on) and the font to use for footntoes. However, for some projects the user may wish to use a specific format that differs to the preferences he or she has set up for most projects. Of course, in Scrivener the user could just choose to override the format at the Compile Draft stage, but some users prefer to write in the output font. For these users, Scrivener 2.0 provides per-project "Text Formatting Preferences" (available via the Project menu). These settings allow the user to override the text formatting and footnote font set in the general Scrivener preferences for a single project. For instance, a user may have his Preferences set up so that when typing in new documents, the document uses an Optima 12-point font and double line spacing, and footnotes are typed in a 10-point font. For a specific project, however, he may wish to use Garamond 13-point with single line spacing and Garamond 11-point for footnotes. It would be inconvenient to have to keep changing the preferences when switching between projects, and equally inconvenient to change the formatting in each new document created in the project. In this case, the user could choose to add these Garamond settings as project preferences so that they override the settings in the general preferences. This is the Text Formatting Preferences sheet in Scrivener 2.0:




Although general preferences are stored at the OS-level (and thus will be handled differently on different platforms), the project preferences are stored within the project package. They are stored in the file **projectpreferences.xml**, which is kept inside the .scriv/Settings folder (if there are no project preferences, this file may not exist).

A project preferences file contains the following information:

- Whether the project preferences should override the general preferences or be ignored.
- The text formatting to use when typing in new documents if project preferences are set to override the general preferences.
- Whether footnotes should use a different font.
- The footnote font to use if set to use a different one.

The elements of the **projectpreferences.xml** file are as follows:

## ProjectPreferences Element

**Description:** This is the root element of the projectpreferences.xml file.
**Subelements:** **UseProjectPreferences, TextFormatRTFData, UseCustomFootnotesFont, FootnotesFont**
**Attributes:** **Version**
**Number:** Exactly 1.
**Errors:** A projectpreferences.xml file without a **ProjectPreferences** element is an invalid document.
**Value:** None.

### Version attribute

**Description:** The **Version** attribute is currently ignored but may be used in the future if the format of the projectpreferences.xml file changes.
**Required:** No.
**Value:** "1.0".

## UseProjectPreferences Element

**Description:** Determines whether the settings defined in the projectpreferences.xml file should be used instead of those set in the general preferences, or whether they should be ignored.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If no **UseProjectPreferences** element is present, it should be assumed that the general preferences should take precedence.
**Value:** "Yes" or "No"

## TextFormatRTFData Element

**Description:** Encodes the text formatting that should be used by new documents created in the project should use as RTF data wrapped in hexadecimal.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If no **TextFormatRTFData** element is present, the program should provide a default format.
**Value:** A string of RTF encoded as hexadecimal data. To encode the formatting, Scrivener creates a placeholder string (Scrivener for the Mac uses the arbitrary string "Attributes", but any string of one or more characters will do) and formats it using the specified text attributes, then encodes this rich text string in RTF format. The RTF data object is then converted to hexadecimal and the resulting hexadecimal string is used as the value of the **TextFormatRTFData** element. When reading out the formatting, Scrivener reads the hexademical string, converts it to RTF data, reads this into a rich text string, and grabs the formatting of the first character (which is why it doesn't matter what string is used).

## UseCustomFootnotesFont Element

**Description:** Determines whether a different font should be used for footnotes (instead of the one set in **TextFormatRTF**). Only applicable if **UseProjectPreferences** is set to "Yes".
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1. If this element is not present, assume "No".
**Value:** "Yes" or "No"

## FootnotesFont Element

**Description:** Encodes the font to use for footnotes typed in Scrivener.
**Subelements:** None.
**Attributes:** **Name**, **Size**
**Number:** Exactly 1. If this element isn't present, the program should provide a default font.
**Value:** None.

### Name attribute

**Description:** The **Name** attribute encodes the name of the font.
**Required:** Yes.

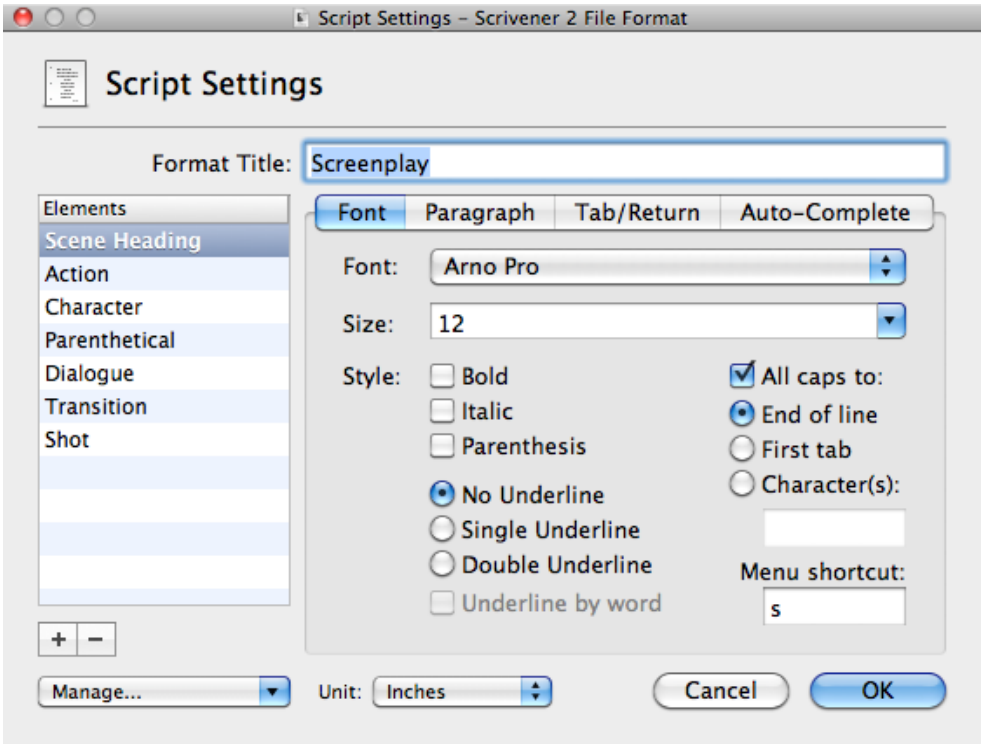
**Value:** The full PostScript name of the font, e.g. “helvetica-oblique”.

### Size attribute

**Description:** The **Size** attribute encodes the size of the font in points.
**Required:** Yes.
**Value:** The size of the font as a floating point number, e.g. “12.0”.

# *14. The Script Format XML File*

Scrivener's script format settings can be set up using the Script Settings panel available via Text > Scriptwriting > Script Settings:




Script settings are encoded into XML and can be stored in the following places:

• If a user creates a custom script format, it will be stored inside the .scriv/Settings folder (i.e. Inside the project itself) using the file name scriptformat.xml.
• If the user chooses to make the script format available to other projects (by selecting "Save for use with other projects" from the "Manage…" button at the bottom of the Script Settings panel), an identical XML file will be saved inside the ~/Application Support/Scrivener/ ScriptFormats folder on Mac OS X - this is the the folder Scrivener for OS X looks in for script formats that can be selected from the Format > Scriptwriting menu.
• In Scrivener 2.0, some script format XML files are embedded into the application itself, which will also be available via the Format > Scriptwriting menu.

In all cases, the elements of the script format XML files are as follows:

## ScrivenerScriptFormat Element

**Description:** This is the root element of a script format XML file.
**Subelements:** **Name, Units, ScriptElements**
**Attributes:** **Version**
**Number:** Exactly 1.
**Errors:** A script format XML file without a ScrivenerScriptFormat element is an invalid document.

**Value:** None.

### Version attribute

**Description:** The **Version** attribute is currently unused but may be used in the future if the file format changes.
**Required:** No.
**Value:** "1.0".

## Name Element

**Description:** The **Name** element encodes the name of the script format which will be used in, for instance, the Scriptwriting menu (e.g. "Comic Book Script" or "Cole & Haag Script").
**Subelements:** None
**Attributes:** None
**Number:** Exactly 1.
**Errors:** If no **Name** element is present, the script format will default to being titled simply "Screenplay".
**Value:** A string containing the name of the script format.

## Units Element

**Description:** The **Units** element encodes the name of the measurement units used for margins and indents in the current script format (centimetres, inches or points).
**Subelements:** None
**Attributes:** None
**Number:** Exactly 1.
**Errors:** If no **Units** element is present, inches will be used as the measurement unit by default.
**Value:** "Inches", "CM" or "Points".

## ScriptElements Element

**Description:** The **ScriptElements** element encodes all of the script elements and their settings and formatting for the current script format - e.g., what paragraph style should be used for "Scene Heading", "Action", etc, what should happen when the user hits tab or return after the current paragraph and so on.
**Subelements:** **Element**
**Attributes:** None
**Number:** Exactly 1.
**Errors:** Because this element stores the actual script elements and their

formatting (“Scene Heading”, “Action” etc), this element must be present for a script format to work properly - without it, there will be no formatting or script elements to use when the user switches into script mode.

**Value:** None.

## Element Element

**Description:** The **Element** element encodes a single script element (such as “Scene Heading” or “Dialogue”) and its associated formatting and settings.

**Subelements:** **Title, KeyEquivalent, CharacterFormat, ParagraphFormat, NextElement, AllowTabs, NextElementOnEmptyLineTab, TabbingAfterTypingBehavior, NextElementOnTabAfterTyping, TextToInsertOnTab, AutoCompleteList**

**Attributes:** None

**Number:** There should be as many **Element** elements as there are script elements for the current script format.

**Errors:** None.

**Value:** None.

## Title Element

**Description:** The **Title** element encodes the title of a script element (e.g. “Scene Heading”, “Transition”).

**Subelements:** None.

**Attributes:** None.

**Number:** Exactly 1 in each **Element** element.

**Errors:** If no **Title** element is present, Scrivener gives the script element a default title of “Element”.

**Value:** A string containing the title of the current script element (e.g. “Scene Heading”, “Parenthetical”).

## KeyEquivalent Element

**Description:** The **KeyEquivalent** element encodes a single character that can be used as a keyboard shortcut for selecting the script element from a menu in Scrivener.

**Subelements:** None.

**Attributes:** None.

**Number:** Exactly 1 in each **Element** element.

**Errors:** None. (No keyboard shortcut is assigned if no **KeyEquivalent** element is present.)

**Value:** A single character that can be used as a keyboard shortcut for

the script element in a menu in Scrivener. In Scrivener for Mac OS X, for instance, this keyboard shortcut is used in the script elements menu in the footer bar at the bottom of the editor when in script mode.

## CharacterFormat Element

**Description:** The **CharacterFormat** element encodes the character style to use for the current script element, such as the font and underline style.
**Subelements:** **Font, UnderlineStyle, AllCaps, StyleDelimiter, TextColor.**
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Errors:** If no **CharacterFormat** element is present, Scrivener will default to using a 12-point Courier font with no other attributes.
**Value:** None.

## Font Element

**Description:** The **Font** element encodes the font to be used for the current script element.
**Subelements:** None.
**Attributes:** **Name, Size**
**Number:** Exactly 1 in each **CharacterFormat** element.
**Errors:** If the **Font** element is missing, or if the selected font is not available on the current system, Scrivener defaults to a Courier font or to the default user font (e.g. Helvetica on OS X) if no Courier font is available.
**Value:** None.

### Name attribute

**Description:** The **Name** attribute of the **Font** element encodes the full font name as used in PostScript.
**Required:** Yes.
**Value:** The full font name, as used in PostScript language code, e.g. “Times-Roman”. (Note, however, that the **Bold** and **Italic** subelements of the **CharacterFormat** element will override any italic or bold settings contained in the font name.)

### Size attribute

**Description:** The **Size** attribute of the **Font** element encodes the font size in points.
**Required:** Yes. (If not present, defaults to 12.)
**Value:** The font size in points as a floating point number to one

decimal place, e.g. “12.0”.

## Bold Element

**Description:** The **Bold** element encodes whether the font of the current script element should be bold or not.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **CharacterFormat** element. If not present, it should be assumed that the font is not bold.
**Errors:** None.
**Value:** “Yes” or “No”.

## Italic Element

**Description:** The **Italic** element encodes whether the font of the current script element should be italicised or not.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **CharacterFormat** element. If not present, it should be assumed that the font is not italic.
**Errors:** None.
**Value:** “Yes” or “No”.

## UnderlineStyle Element

**Description:** The **UnderlineStyle** element encodes the underline style of the current script element.
**Subelements:** None.
**Attributes:** **ByWord**.
**Number:** Exactly 1 in each **CharacterFormat** element. (If not present, it should be assumed that the text is not underlined.
**Errors:** None.
**Value:** “Single”, “Double” or “None”.

## ByWord attribute

**Description:** The **ByWord** attribute of the **UnderlineStyle** element determines whether or not the underline style only appears beneath words, and not beneath white space such as tabs and spaces.
**Required:** No. If not present, then regular underlining is used (*not* “by word”).
**Value:** “Yes” or “No”.

## AllCaps Element

**Description:** The **AllCaps** element encodes whether or not the current script element uses all uppercase characters.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **CharacterFormat** element. (If not present, it should be assumed that the text is not in all uppercase.)
**Errors:** None.
**Value:** "Yes" or "No".

## TextColor Element

**Description:** The **TextColor** element encodes the font colour used by the current script element.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **CharacterFormat** element. (If not present, it should be assumed that the text is black.)
**Errors:** None.
**Value:** Three floating point numbers representing the red, green and blue components of the colour, separated by a space. The red, blue and green components should be a fraction of 1, and will thus range between 0.0 and 1.0. For instance, 1.0 0.0 0.0 would be red, 0.5 0.5 0.5 would be grey and 0.0 0.5 0.0 would be dark green. Note that the colour will be ignored for the "No Label" item (which always has an internal ID of −1), which has no colour associated with it in Scrivener's interface.

## StyleDelimiter Element

**Description:** The **StyleDelimiter** element encodes text at which the script element ceases to use uppercase, bold, italic and underline formatting. For instance, if the **CapsDelimiter** element value is a colon, then all text for the current script element will appear in uppercase, bold or whatever styling is set until a colon is typed, after which any typing will appear using normal case and a plain font. This is useful for formats such as UK stage play format, where character names are capitalised, then there is a tab, and then dialogue is written in normal lower- and uppercase characters.
**Subelements:** None.
**Attributes:** None.

**Number:** 1 or 0 in each **CharacterFormat** element. If there is no **StyleDelimiter** element, or if its value is empty, it should be assumed that the script element should use the set style (bold, uppercase etc) up to the end of the line.
**Errors:** None.
**Value:** Any string of characters surrounded by brackets. Note that the brackets are ignored and are only added to ensure that a whitespace delimiter is not stripped by the XML read-write processes. E.g. A value of "(:)" indicates that a single colon should be used as the caps delimiter.

## ParagraphFormat Element

**Description:** The **ParagraphFormat** element encodes the paragraph formatting, such as line spacing and indents, of the current script element.
**Subelements:** **Parenthetical, KeepWithNext, Alignment, Spacing, SpaceBefore, FirstLineIndent, Indent, TailIndent, FirstTab, DefaultTabInterval, WritingDirection, MinLineHeight, MaxLineHeight**
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Value:** None.

## Parenthetical Element

**Description:** The **Paranthetical** element encodes whether the current script element should be enclosed in brackets. Scrivener automatically adds brackets around such elements, just as Final Draft, Movie Magic Screenwriter etc do when formatting "Parenthetical" script elements.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **ParagraphFormat** element. If not present, the script element should not be treated as a parenthetical element.
**Value:** "Yes" or "No".

## KeepWithNext Element

**Description:** The **KeepWithNext** element encodes whether the script element should be kept together with the following paragraph. For example, when writing a screenplay, it is standard practice to ensure that "Character" names do not get stranded at the end of a page so that they are separated from the dialogue that follows them. In this case, if **KeepWithNext** is set to "Yes" for the

| | |
|---|---|
| | "Character" element, the character name would get pushed onto the next page to keep it together with its associated dialogue. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0 in each **ParagraphFormat** element. If not present, keep-with-next is turned off. |
| **Value:** | "Yes" or "No". |

## Alignment Element

| | |
|---|---|
| **Description:** | The **Alignment** element encodes the text alignment of the current script element. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0 in each **ParagraphFormat** element. If not present, assume left text alignment. |
| **Value:** | "Left", "Justified", "Right" or "Natural". Note that "Natural" uses left or right alignment depending on the user's language settings (e.g. English would use left alignment, Hebrew would use right). |

## Spacing Element

| | |
|---|---|
| **Description:** | The **Spacing** element encodes the spacing used between lines in paragraphs of the current script element (e.g. double line spacing or single line spacing). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1 in each **ParagraphFormat** element. If not present, assume single line spacing. |
| **Value:** | "1.0" for single line spacing, "1.5", or "2.0" for double line spacing. |

## SpaceBefore Element

| | |
|---|---|
| **Description:** | The **SpaceBefore** element encodes the line spacing that should be inserted before paragraphs of the current script element. For instance, if line spacing is "1", there should be a gap the height of a single line between the previous paragraph and the current one. Note that the height of the spacing is calculated based on the current font height, so if the current script element has a "space before" value of 2 and uses a 12 point font, then the actual line spacing inserted before the paragraph will be 24 |

| | |
|---|---|
| | points high. Also note that no actual extra lines are inserted, but only padding. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1 in each **ParagraphFormat** element. If not present, assume no space before. |
| **Value:** | 0, 1, 2 or 3. |

## FirstLineIndent Element

| | |
|---|---|
| **Description:** | The **FirstLineIndent** element encodes how far from the left margin the first line of the paragraph will be indented. It uses the units set in the **Units** element of the script format (see above). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1 in each **ParagraphFormat** element. If not present, assume no indent. |
| **Value:** | A floating point number representing a distance from the left margin using the measurement unit set in the **Units** element. E.g. If the units are set to points, a value of 36 is the same as a value of 0.5 when the units are set to inches. |

## Indent Element

| | |
|---|---|
| **Description:** | The **Indent** element encodes how far from the left margin all subsequent lines in the paragraph after the first will be indented. It uses the units set in the **Units** element of the script format (see above). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1 in each **ParagraphFormat** element. If not present, assume no indent. |
| **Value:** | A floating point number representing a distance from the left margin using the measurement unit set in the **Units** element. E.g. If the units are set to points, a value of 36 is the same as a value of 0.5 when the units are set to inches. |

## TailIndent Element

| | |
|---|---|
| **Description:** | The **TailIndent** element encodes how far from the left margin the text in the paragraph should run to. For example, if it is set to 6.5 inches, any text in a line that would go beyond 6.5 inches |

| | |
|---|---|
| | will wrap onto the next line in the paragraph. It uses the units set in the **Units** element of the script format (see above). Note that if this number is negative, then the tail indent is measured from the right margin. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1 in each **ParagraphFormat** element. If not present, assume no tail indent (i.e. the text should run right up to the right margin of the page). |
| **Value:** | A floating point number representing a distance from the left margin using the measurement unit set in the **Units** element. E.g. If the units are set to points, a value of 36 is the same as a value of 0.5 when the units are set to inches. The value can be negative to represent the distance from the right margin instead. |

## FirstTab Element

| | |
|---|---|
| **Description:** | The **FirstTab** element encodes the distance from the left margin of the first tab stop, if any. It uses the units set in the **Units** element of the script format (see above). |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | 1 or 0 in each **ParagraphFormat** element. If not present, assume no tabs. |
| **Value:** | A floating point number representing a distance from the left margin using the measurement unit set in the **Units** element. If 0, no tab will be added. |

## DefaultTabInterval Element

| | |
|---|---|
| **Description:** | The **DefaultTabInterval** element encodes the distance between any tabs added to the current script element by Scrivener. It uses the units set in the **Units** element of the script format (see above). If a **FirstTab** element was set, any subsequent tabs will be added after it using the distance set by this element, otherwise the first tab (if any) will be added this distance from the left margin. Normally, Scrivener does not add any tabs at all if **FirstTab** is not set, and does not add any extra tabs after the first tab if **FirstTab** *is* set. However, because Scrivener for the Mac relies on paragraph formatting to recognise different script elements, it is essential that each script element has a different paragraph style. Therefore, if two script elements use exactly the same paragraph style (for instance, "Scene Heading" and "Shot"), Scrivener will add one or more tabs to one of the script |

element paragraph formats just to ensure that they are different. It only uses the **DefaultTabInterval** setting to determine how far apart to place tabs in this circumstance. This setting can safely be ignored in systems using a different approach to recognising script elements in the text.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0 in each **ParagraphFormat** element. If not present, assume 0.5 inch tab intervals if applicable.

**Value:** A floating point number representing the distance between tab stops using the measurement unit set in the **Units** element. E.g. If the units are set to points, a value of 36 is the same as a value of 0.5 when the units are set to inches.

## WritingDirection Element

**Description:** The **WritingDirection** encodes the writing direction to be used in paragraphs of the current script element. Usually the writing direction will be left-to-right, but in some languages it may be set to right-to-left.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0. If not present, assume natural writing direction (that is, the default writing direction for the user's current language settings).

**Value:** "LeftToRight" or "RightToLeft".

## FixedLineHeight Element

**Description:** The **FixedLineHeight** element encodes an exact line height for the current script element. For example, if **FixedLineHeight** is set to 24, all lines for the current script element will be exactly 24 points high, even if the font is only 12 points high. If it is set to 12, all lines for the current script element will be restricted to a minimum and maximum height of 12 points, even if the font is set to 18 points (in this case resulting in some of the text being clipped). Note that **FixedLineHeight** may be necessary to ensure leading is constant between different text systems. The OS X Cocoa text system, for instance, uses a line height of 14 for text with a 12-point font unless the line height is restricted. This can result in layout differences between Scrivener and Final Draft, which uses a 12-point line height for 12-point text. The default scriptwriting format in Scrivener therefore uses a fixed line height of 12 for script elements that use a 12-point

font.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0 in each **ParagraphFormat** element. If not present, assume no fixed line height (i.e. the line height should automatically adjust to fit the font height, using the default behaviour of the current text engine).

**Value:** A floating point number representing the minimum and maximum line height in points. Can be 0 if there is no fixed line height (in which case the line height should automatically adjust to fit the text height).

## NextElement Element

**Description:** The **NextElement** element encodes which element follows the current script element. This is used when the user hits return. For instance, if the user has just typed a "Scene Heading" and hits return, if the **NextElement** of the "Scene Heading" script element is set to "Action", then when the user starts typing again, he or she will be typing an "Action" element.

**Subelements:** None.

**Attributes:** None.

**Number:** Exactly 1 in each **Element** element. If not present, when the user hits return the next element used should be the first element in the script elements list.

**Value:** A string containing the title of the next element to go to.

## AllowTabs Element

**Description:** The **AllowTabs** element determines whether the user can enter tabs in paragraphs of the current element. If set to "Yes", then hitting the tab key acts as it would normally when not in script mode; if "No" then the tab key can be set to use special behaviour such as changing to a different element or inserting some specified text.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0 in each **Element** element. If not present, assume "No".

**Value:** "Yes" or "No".

## NextElementOnEmptyLineTab Element

**Description:** The **NextElementOnEmptyLineTab** element determines which

element to change to if the user hits tab at the start of a blank line (note that this setting is only applicable if **AllowTabs** is set to "No"). For instance, if the user hits return after a "Character" element and **NextElement** was set to "Dialogue" so that the user is now ready to type dialogue, if **NextElementOnEmptyLineTab** is set to "Parenthetical", the user could immediately hit the tab key to change the current element to the parenthetical element and so type a parenthetical instead of dialogue.

**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element. If not present, assume 0 (first element).
**Value:** A string containing the title of the next element to go to when tabbing on an empty line.

## TabbingAfterTypingBehavior Element

**Description:** The **TabbingAfterTypingBehavior** element determines what will happen if the user hits the tab key after having typed some text in the current element (note that this setting is only applicable if **AllowTabs** is set to "No"). There are two things that can happen when a user hits tab after typing: either typing will move to a different element on a new line, or some specified text can be automatically inserted (for instance, hitting tab after a location in a "Scene Heading" element might add a hyphen).
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Value:** "NewElement" to specify that tabbing after typing will move to a new element (the new element will be determined by the **NextElementOnTabAfterTyping** element), or "InsertText" if hitting tab should insert some predefined text (which will be determined by the **TextToInsertOnTab** element).

## NextElementOnTabAfterTyping Element

**Description:** The **NextElementOnTabAfterTyping** element determines which script element typing should move to after hitting a tab if **AllowTabs** was set to "No" and **TabbingAfterTypingBehavior** was set to "NewElement". For instance, a "Character" element may move to the "Dialogue" element when the user hits return (as determined by **NextElement**), but to the "Parenthetical" element when the user hits tab after typing the character name (as determined by **NextElementOnTabAfterTyping**).

**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Value:** A string containing the title of the next element to go to.

## TextToInsertOnTab Element

**Description:** The **TextToInsertOnTab** element specifies text that should be inserted upon hitting the tab key after typing in the current element. It is only applicable if **AllowTabs** was set to "No" and **TabbingAfterTypingBehavior** was set to "InsertText". For instance, a "Scene Heading" element may be set to insert " - " upon hitting tab after typing, so that the user could type, e.g., "INT. BEDROOM" and then hit tab to insert the " - " separator that goes before the time of day.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Value:** A string containing the text that should be inserted on hitting tab, enclosed in brackets. Note that the brackets are ignored and are only added to ensure that text containing only white space is not stripped by the XML read-write processes. E.g. A value of "( - )" indicates that " - " should be inserted upon hitting tab.

## AutoCompleteList Element

**Description:** The **AutoCompleteList** element encodes the auto-complete list specific to the current script element. For example, a "Scene Heading" script element may have an auto-complete list containing "INT.", "EXT.", "DAY", "NIGHT", etc, while a "Transition" script element may have an auto-complete list containing "FADE IN.", "FADE OUT." etc.
**Subelements:** **Completions, AppendProjectAutoCompleteList, ProjectListAutoCompleteBehavior, AutomaticallyAddPhrasesToProjectList, PhraseStartDelimiter, PhraseEndDelimiter**
**Attributes:** None.
**Number:** Exactly 1 in each **Element** element.
**Value:** None.

## Completions Element

**Description:** The **Completions** element encodes a list of words that should be auto-completed for the current script element. For instance, if

| | |
|---|---|
| | "INT." and "EXT." are set as completions for a "Scene Heading" script element, then if the user starts typing "i", "INT." will pop up as a suggested completion. |
| **Subelements:** | **Completion** |
| **Attributes:** | None. |
| **Number:** | 1 or 0 in each **AutoCompleteList** element. If not present, this script element does not have a specific auto-complete list. |
| **Value:** | None. |

## Completion Element

| | |
|---|---|
| **Description:** | The **Completion** element encodes a single auto-completion word or phrase for the current script element. |
| **Subelements:** | None. |
| **Attributes:** | **NewLineAfter** |
| **Number:** | There should be as many **Completion** subelements within the **Completions** element as there are auto-completion words or phrases for the current script element. |
| **Value:** | A string representing the word or phrase to be auto-completed (e.g. "INT.", "TRANSITION"). |

### NewLineAfter attribute

| | |
|---|---|
| **Description:** | The **NewLineAfter** attribute of the **Completion** element determines whether or not typing should move to a new line after auto-completing the word encoded by the **Completion** element. For instance, this would be set to "No" for phrases or words that would normally be auto-completed at the start of a line such as "INT." and "EXT." in scene headings, whereas it would normally be set to "Yes" for phrases or words that usually end a line, such as "DAY" or "NIGHT" in scene headings. |
| **Required:** | No. If not present, then assume "No". |
| **Value:** | "Yes" or "No". |

## AppendProjectAutoCompleteList Element

| | |
|---|---|
| **Description:** | The **AppendProjectAutoCompleteList** determines whether words from the project auto-complete list (the user-defined list of auto-completions in any Scrivener project set up via Project > Edit Auto-Complete List) should also be suggested during typing for the current script element. For instance, if the user has "Ian" in the project auto-complete list and "INT." as one of the completions for the current script element, if he types "i", then both "INT." and "Ian" would be suggested if |

**AppendProjectAutoCompleteList** is set to "Yes", whereas only "INT." would be suggested if it is set to "No".

**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **AutoCompleteList** element. If not present, assume "No".
**Value:** "Yes" or "No".

## ProjectListAutoCompleteBehavior Element

**Description:** The **ProjectListAutoCompleteBehavior** element determines what should happen when the user hits return to auto-complete a word from the project auto-completion list when typing in the current script element.
**Subelements:** None.
**Attributes:** None.
**Number:** 1 or 0 in each **AutoCompleteList** element. If not present, there is no special behaviour - the user just carries on typing.
**Value:** "GoToNextLine", "InsertTab" or "Default". If "GoToNextLine" is set, then upon hitting return to complete a word from the project list, typing will move to the next line and script element (this is the same as setting **NewLineAfter** to "Yes" for the script element completions, except that this only affects the behaviour when completing words from the project list). If "InsertTab" is set, then a tab is inserted after the auto-completed word after accepting the completion. Note that this is useful, for instance, when setting up a play format in which tabs separate character names and dialogue. In this case, the user may start typing a character name, see the character name pop up as a possible completion from the project list, accept the completion, and then have a tab automatically inserted so that the user is immediately ready to start typing dialogue. If "Default" or any other value is set, then nothing special will happen after accepting a project auto-completion.

## AutomaticallyAddPhrasesToProjectList Element

**Description:** The **AutomaticallyAddPhrasesToProjectList** element determines whether anything typed in the current element should be automatically added to the project auto-complete list (the list the user can set up for any Scrivener project by going to Project > Edit Auto-Complete List). This is useful, for instance, for "Character" script elements, where a user may wish any new character names typed to be automatically added to the project

auto-complete list. This setting can be refined using the **PhraseStartDelimiter** and **PhraseEndDelimiter** elements, which allow only phrases between certain characters to get detected and added to the list. For instance, the format of a standard screenplay scene heading is "INT./EXT. LOCATION - TIME OF DAY" - e.g., "INT. NAKOTAMI PLAZA - EVENING". In this case, the user may want any locations he or she types to be automatically detected and added to the project auto-complete list. By setting **AutomaticallyAddPhrasesToProjectList** to "Yes", **PhraseStartDelimiter** to "." and **PhraseEndDelimiter** to "-", Scrivener can detect the location and add it to the project auto-complete list by just looking at the text that appears between the period and the hyphen. (Note that Scrivener adds such phrases to the project auto-complete list when the user goes to the next line.)

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0 in each **AutoCompleteList** element. If not present, phrases are *not* automatically added to the project auto-complete list.

**Value:** "Yes" or "No".

## PhraseStartDelimiter Element

**Description:** The **PhraseStartDelimiter** element encodes the first text snippet that Scrivener will look for when auto-detecting words or phrases to add to the project auto-complete list. This element is ignored if **AutomaticallyAddPhrasesToProjectList** is set to "No". For example, if **PhraseStartDelimiter** is set to "-", then when the user moves to the next line, Scrivener will scan the text of the current element looking for a hyphen, and anything after the hyphen (up to the text set in **PhraseEndDelimiter**) will be added to the project auto-complete list if it isn't already there. If there is no **PhraseStartDelimiter** element, or if it is blank, phrases will be added from the start of the line.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0 in each **AutoCompleteList** element. If not present, phrases and words will be added to the project auto-complete list from the start of the paragraph (but only if **AutomaticallyAddPhrasesToProjectList** is set to "Yes" - otherwise it is ignored).

**Value:** A string containing the text of the delimiter enclosed in brackets. Note that the brackets are ignored and are only added to ensure that a whitespace delimiter is not stripped by the

XML read-write processes. E.g. A value of "(.)" for **PhraseStartDelimiter** and a value of "(-)" for **PhraseEndDelimiter** tells Scrivener to add any words or phrases that occur between a period and a hyphen to the project auto-complete list.

## PhraseEndDelimiter Element

**Description:** The **PhraseEndDelimiter** element encodes a text snippet marking the end of sections of text within the current script element that should be automatically added to the auto-complete list. Scrivener looks for this text when auto-detecting words or phrases to add to the project auto-complete list if **AutomaticallyAddPhrasesToProjectList** is set to "Yes". For instance, if **PhraseEndDelimiter** is set to ":", then when the user moves to the next line, Scrivener will scan the text of the current element looking for a colon, and anything up to the colon (but after the text set in **PhraseStartDelimiter**, if any) will be added to the project auto-complete list if it isn't already there. If there is no **PhraseEndDelimiter** element, or if it is blank, Scrivener will scan right up to the end of the paragraph when looking for new words or phrases to add. E.g. If **AutomaticallyAddPhrasesToProjectList** is set to "Yes" but both **PhraseStartDelimiter** and **PhraseEndDelimiter** are both either blank or missing, then the entire paragraph will get added as a single auto-complete phrase to the project list.

**Subelements:** None.

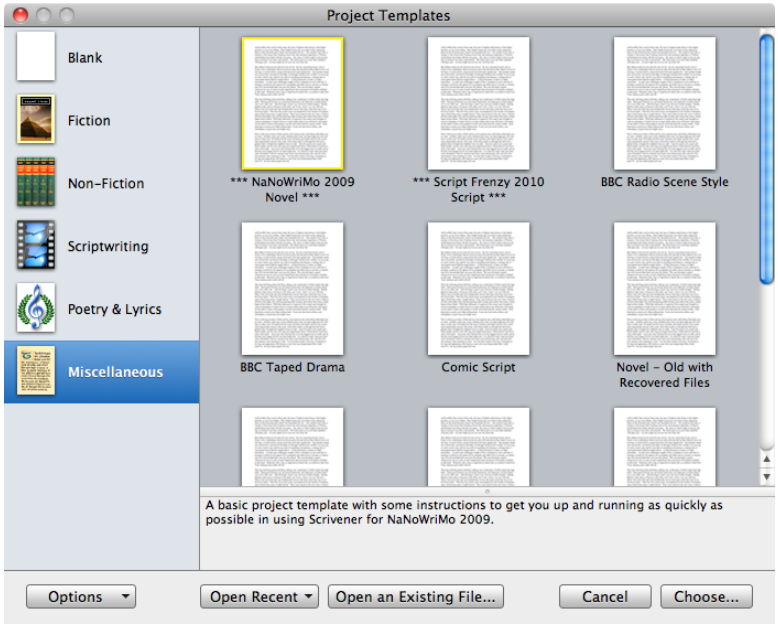
**Attributes:** None.

**Number:** 1 or 0 in each **AutoCompleteList** element. If not present, phrases or words will be added to the project auto-complete list up to the end of the paragraph (but only if **AutomaticallyAddPhrasesToProjectList** is set to "Yes" - otherwise it is ignored).

**Value:** A string containing the text of the delimiter enclosed in brackets. Note that the brackets are ignored and are only added to ensure that a whitespace delimiter is not stripped by the XML read-write processes. E.g. A value of "(.)" for **PhraseStartDelimiter** and a value of "(-)" for **PhraseEndDelimiter** tells Scrivener to add any words or phrases that occur between a period and a hyphen to the project auto-complete list.

# *15. Project Templates*

Project templates are used to create new projects from within Scrivener. They appear in a templates chooser panel when the user chooses File > New Project:




Scrivener comes with a number of project templates "out-of-the-box", including ones for writing novels that get exported to standard manuscript format, screenplays, stage plays, comics and so on. The user can create his or her own templates by creating a skeletal Scrivener project that contains only what he or she wants to appear in new projects when created from the template, and then selecting File > Save As Template.

Scrivener template files are saved as XML files on disk. When the user opens the project templates chooser, Scrivener looks in preset locations for .scrivtemplate files and uses information stored in these XML files to populate the templates chooser. When a user creates a project from a template, or saves a project as a template, certain information about the template - its title, description and icon - are also saved inside the project itself, so that it can be used to populate the Save As Template sheet if necessary.

(Note: only the "Blank" option in the templates chooser does not have an underlying template file associated with it - if the user selects "Blank", a .scriv package is created on disk which contains only the bare minimum files Scrivener needs in order to run, and a Scrivener project window is opened which contains just the Draft, Research and Trash folders, with a single text document inside the Draft folder selected and ready to edit.)

# *15.1. The .scrivtemplate XML File Format*

Project templates in Scrivener 2.0 are XML files that have the .scrivtemplate file extension.

On the Mac version, project templates are stored in two places: project templates provided by Scrivener are stored inside a Default Support Folder inside the Scrivener application bundle itself; user-generated templates are stored inside the ~/Library/Application Support/Scrivener/ProjectTemplates folder. (On Windows, templates provided by the program may be placed inside Scrivener's Program Files directory, perhaps.)

Project template files have the file extension "scrivtemplate" and encode an entire project, which can be used as a basis for creating new projects, along with some information about it. Specifically, a project template file encodes the following information:

- The title of the template.
- A description which will provide the user with more information about what is contained in the template when it is selected in the templates chooser panel.
- An icon that represents the template. This is usually an image of a page that gives the user an idea about what documents might be created using the template, sometimes with a small icon in its corner if the page on its own would be ambiguous. Scrivener provides some default icons, so a template file may encode either the name of a preset or the image data of a custom icon.
- The data that will be used to create the project itself. The project data is encoded as zip data represented in hexadecimal format. When a new project is created from a template, this hexadecimal data is used to create a zip file on disk from which the new project is then extracted.

The contents of a .scrivtemplate XML file are as follows:

## ScrivenerTemplate Element

| | |
|---|---|
| **Description:** | This is the root element of a .scrivtemplate file. |
| **Subelements:** | **Title, Description, Category, ImageName, CustomImageData, ProjectZipData** |
| **Attributes:** | **Version** |
| **Number:** | Exactly 1. |
| **Errors:** | A .scrivproject file without a **ScrivenerTemplate** element is an invalid document. |
| **Value:** | None. |

## Version attribute

| | |
|---|---|
| **Description:** | The **Version** attribute is currently unused but may be used in the future if the format of .scrivproject files changes. |
| **Required:** | No. |
| **Value:** | "1.0". |

## Title Element

**Description:** The **Title** element encodes the title of the project template.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Value:** A string containing the title of the project template.

## Description Element

**Description:** The **Description** element encodes text describing the project template. When a user clicks on a template, this text appears in the template chooser panel to provide the user with information about what the template contains.
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Value:** A string containing a description of the project template.

## Category Element

**Description:** The **Category** element encodes the category to which the project template belongs. (In Scrivener 2.0, templates may be assigned to several different categories such as Fiction, Non-Fiction etc.)
**Subelements:** None.
**Attributes:** None.
**Number:** Exactly 1.
**Errors:** If not present, or if Scrivener doesn't recognise the value as one of the predefined category values, Scrivener will place the template in the "Miscellaneous" category.
**Value:** A string containing one of the following values: "Blank", "Fiction", "NonFiction", "Scriptwriting", "Poetry" or "Miscellaneous".

## ImageName Element

**Description:** The **ImageName** element encodes the name of the image used to represent the project template in Scrivener's New Project template chooser.
**Subelements:** None.
**Attributes:** None.

**Number:** 1 or 0. If the **ImageName** element is not present, then the **CustomImageData** should be. (Templates can use either an image provided by Scrivener, in which case the **ImageName** will be set, or a custom image provided by the user, in which case the custom image will be encoded in **CustomImageData**).

**Errors:** If the **ImageName** does not correspond to one of the predefined values, Scrivener will use "TemplateGenericText" (a generic text icon).

**Value:** A string containing one of the following values, each of which corresponds to an image inside Scrivener that will be used to represent the template: "TemplateBlank", "TemplateNovel", "TemplateShortStory", "TemplateAcademic", "TemplateAcademicWithIcon", "TemplateScreenplay", "TemplateScreenplayWithIcon", "TemplateStageplay", "TemplatePlayUK", "TemplatePlayUS", "TemplateComic", "TemplateRadio", "TemplateRadioAlternative", "TemplateTelevision", "TemplateSitCom", "TemplatePoetry", "TemplateGenericText".

## CustomImageData Element

**Description:** The **CustomImageData** element encodes in hexadecimal format an image provided by the user to be used to represent the project template in the New Project templates chooser panel. This element can be used instead of **ImageName** for providing custom images not built into Scrivener.

**Subelements:** None.

**Attributes:** None.

**Number:** 1 or 0. The **CustomImageData** element only exists if a custom image needs to be encoded to represent this template. If the **CustomImageData** element isn't present, then the **ImageName** element must be present to specify one of the provided images. That is, **CustomImageData** and **ImageName** should be mutually exclusive. **ImageName** specifies the name of an image already built into Scrivener. But if the user has created his or her own image which he or she wants to use to represent the template, then it has to be encoded into the template file itself using the **CustomImageData** element so that Scrivener can recreate the image for use in the templates chooser panel.

**Errors:** If neither **IconName** nor **CustomImageData** are present, Scrivener will default to using the generic text icon to represent a template. If both are present, Scrivener uses **IconName**.

**Value:** A string representing the image encoded as tif data in hexadecimal format.

## ProjectZipData Element

| | |
|---|---|
| **Description:** | The **ProjectZipData** element encodes the .scriv project to be used as the basis of the template in zipped, hexadecimal format. When a user saves a project as a template, Scrivener creates a temporary .zip file containing the .scriv project, and then encodes the .zip file data in hexadecimal format to save into the template. When a user creates a new project from a template, Scrivener creates a temporary .zip file from this hexadecimal data, and then unarchives the temporary .zip file and creates the new project by copying (and renaming) the extracted .scriv file to the location specified for the new project by the user. |
| **Subelements:** | None. |
| **Attributes:** | None. |
| **Number:** | Exactly 1. |
| **Errors:** | If the **ProjectZipData** element does not exist, the .scrivtemplate file is invalid because no new project can be created from it. |
| **Value:** | A string representing the a zipped-up copy of the project from which the template was created encoded in hexadecimal format. |

## *15.2. The templateinfo.xml File Format*

If a project has been created from a template, it remembers information about the template from which it was created by saving this information in a **templateinfo.xml** file inside the .scriv/Settings folder. This information is only ever used by the File > Save As Template sheet, so that the Save As Template sheet can be prefilled with the name, description and icon of the template from which the project was created.

Likewise, if a user uses the Save As Template sheet, upon saving the template, the **templateinfo.xml** file inside the project from which the template was created will be updated (or created if it did not previously exist) to remember the new information.

A Scrivener project is perfectly valid if no **templateinfo.xml** file exists - it is just stored as a convenience to the user so that he or she does not have to re-type information about a template (for instance, when updating an existing template by creating a new project from it and re-saving it over the original template).

Note that the XML contained inside a **templateinfo.xml** file is idential to that stored inside a .scrivtemplate file with the following differences:

1. The **ScrivenerTemplate** root element is named **TemplateSettings** instead.
2. The ProjectZipData element is not present in the **templateinfo.xml** file (because the **templateinfo.xml** file is just used to prefill the fields in the Save As Template sheet with the template title, description and icon).

# 16. Adding Custom Data to Scrivener Projects

It is possible for third-party applications, or versions of Scrivener on different platforms, to save custom data inside a Scrivener project and have it preserved when the project is opened and saved by Scrivener. There are two ways of doing this, which may be used separately or in conjunction: the data can be stored as a custom XML element or elements inside the SCRIVX file itself, or it can be stored in custom files inside a subdirectory of the .scriv package.

## 16.1. Encoding Custom Data in the SCRIVX File

There are two ways to have Scrivener preserve custom project data, depending on whether the data you want to encode applies to the whole project or just to a specific binder item.

To encode project-level information, write the data is the SCRIVX file as one or more elements inside a **UserProjectData** element. The **UserProjectData** element should be a direct sub-element of the **ScrivenerProject** element. Custom elements can have as many attributes or subelements as you require. There are no restrictions on the names of the subelements (it is, however, recommended that namespaces are used to differentiate different sets of third-party custom data to reduce the risk of conflicting names). Scrivener remembers all such custom XML elements, including their attributes and subelements, so that when you open and save the file in Scrivener, all of those elements will still be in the newly-saved file. It doesn't matter how many changes you make to the Scrivener project between opening and saving, as all of the third-party data is preserved exactly. You can add any number of such elements and they can have any structure of subelements and any number of attributes, as well as a value.

For example, in the SCRIVX snippet below, an application called "MyApplication" has added some custom data, using its own application name as the element name under which it has saved two pieces of custom data as the elements **MyCustomString** and **MyCustomDate**:

```
<ScrivenerProject Version="1.0">
   ...Scrivener's subelements...

   <UserProjectData>
      <MyApplication>
         <MyCustomString>Some text.</MyCustomString>
         <MyCustomDate>2010-04-12 21:48:10 +0100</MyCustomDate>
      </MyApplication>
   </UserProjectData>

</ScrivenerProject>
```

To encode information that pertains to individual binder documents, use the **UserData** subelement of **BinderItem** in exactly the same way.

## 16.2. Creating Custom Files in Subfolders

Because Scrivener uses a package (folder) file format, it is also possible for custom data of any type to be stored inside the .scriv directory. Third parties may therefore store custom XML files or files of any other type inside the .scriv package. Scrivener will not remove or overwrite any files added to the package that it does not know about, so they will be left intact and preserved exactly no matter how many times the project is opened and saved.

To ensure such files do not clash with existing files that Scrivener *will* read and change, it is suggested that third-party data is stored inside the Files subdirectory in a custom directory entitled "CustomUserData". For instance, in the following example, an application entitled "MyApplication" has saved a custom file entitled "customdata.xml" into the project package by creating a subdirectory called "CustomUserData" inside the Files folder, and then another subdirectory inside that to store its custom files: