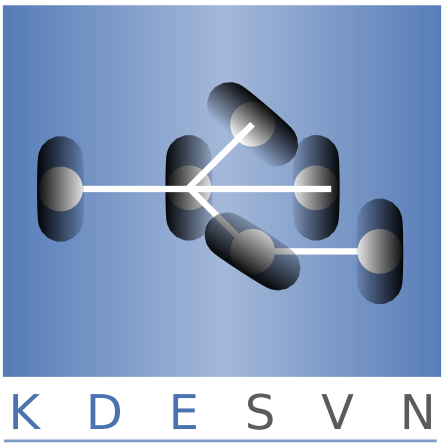
The kdesvn Handbook

Rajko Albrecht



Contents

List of Tables

Abstract

kdesvn - a subversion client by KDE.

Chapter 1

Introduction

kdesvn is a subversion client by KDE.

You should have some knowledge about subversion itself, but hopefully most items are self explanatory.

You may send bug reports and feature wishes via KDE Bugtracking System.

1.1 Terms

If you're familiar with revision control systems you may skip that - or read and correct the author ;)

Repository

Central store of data. That may be a database or a flat filesystem. Without special clients you're not able to read data in it. For Subversion repositories kdesvn is such a client.

Working copy

A flat copy of a repository on local filesystem. This is used like any normal filemanager, editing files and so forth. RCS-information you can read with clients like kdesvn.

Remember that subversion does not know anything about KIO, so a working copy must reside in an area where it may reached without any specific protocol, e.g. 'fish://' or such is not possible.

WebDav

WebDav is a protocol which let you modify files on a remote webserver. Subversion is a special kind of WebDAV when repositories are accessed via webserver. In normal use this is read-only. With special configurations you may get a read-write enabled WebDAV which you may access via specialized browser. kdesvn is NOT a webdav-client, but Konqueror is via the 'webdav://' protocol. But with kdesvn you may browse through the version tree of a repository (via 'http://')

Chapter 2

Using kdesvn

2.1 kdesvn features

kdesvn understands following protocols for browsing repositories:

http
: Standard web browser protocol.

https
: Standard encrypted web browser protocol.

(k)svn+http
: Standard web browser protocol. May be used to let Konqueror automatic call kdesvn.

(k)svn+https
: Standard encrypted web browser protocol. May be used to let Konqueror automatic call kdesvn.

(k)svn+file
: Local repository protocol. May be used to let Konqueror automatic call kdesvn.

(k)svn
: Subversion's own server protocol.

(k)svn+ssh
: Subversion over ssh.

file
: Direct repository access. kdesvn checks if a given path is a repository or a working copy and opens it in the right mode. For subversion `file:// /dir` and `/dir` is not the same!

This list may be used for URLs given for command line, too.

2.2 Beginning with subversion and kdesvn

This section is mostly for beginners not familiar with subversion and explains how subversion and/or revision control systems (RCS) works.

2.2.1 Creating a working copy

> CAUTION
> Working copies MUST be accessible via local paths. Subversion does not know anything about pseudo file systems like `smb://` or `fish://` . kdesvn translates some of them if possible (like `system:/home`) but it is not possible over a network.

First of all you must create a working copy of your repository. For that select **Subversion** → **General** → **Checkout a repository**. Inside the following dialog you must select the URL of the repository you want to use, e.g., something like `http://localhost/repos/myproject`. Subfolders of a repository are possible, too, e.g., `http://localhost/repos/myproject/trunk` or similar.

Select and/or create a local folder, where the working copy should reside.

Last but not least, the revision to checkout. Mostly that should be 'HEAD'. This ensures that the last stored version is the referenced version.

After clicking on **OK** kdesvn will create your new working copy and (if the box was checked) opens it.

When you have opened a repository for browsing you may mark a folder and than select **Subversion** → **Repository** → **Checkout current repository path** and fill out the dialogs as described above. Then only the marked path will checked out.

2.2.2 Committing local changes

Mark the entry or entries you want to send and select **Subversion** → **Working copy** → **Commit** (**Ctrl+#**)

If you try to commit if no item is selected, kdesvn uses the top-most element of opened working copy, i.e., the working copy path itself.

This operation is always recursive, means, if selecting a folder kdesvn always send all changed items below it. When you setup that you want to review all items before commit inside the following dialog all items kdesvn would send are listed. There you may unmark items you don't want to send. In that case kdesvn sends all items alone, i.e., not recursive. Or you may select items not versioned for add and commit (if them are not marked to be ignored). So you may see if there are newer items you forgot.

Enter a log message what you're sending and hit **OK** and the transfer starts.

2.2.3 Update working copy

This brings you local working copy in sync with the repository. You may setup that kdesvn checks on opening a working copy for newer/modified items in repository. This runs in background and you may work meanwhile with kdesvn. When this is finished, items with newer versions or folders with items where newer items are below are marked.

To retrieve the changes, select **Subversion** → **Working copy** → **Update to head**. This will update to the most latest version. If you want to get a specific revision select **Subversion** → **Working copy** → **Update to revision** and select the revision you want in the following box.

If no item is selected, the update will done on whole opened working copy, otherwise only recursive on the selected items.

2.2.4 Adding and Deleting from working copy

Both operations requires two steps: First add or delete and then committing that changes to repository. Before you commit you may undone add or delete.

2.2.4.1 Add items

Adding items into a working copy may be done due three ways:

Select not versioned items and add them
: Copy with Konqueror or any other tool into working copy area. Go trough list, mark them and select **Subversion** → **Working copy** → **Add selected files/dirs** (**Insert**). When you want adding new folders with all subitems select **Subversion** → **Working copy** → **Add selected files/dirs recursive** (**Ctrl+Insert**).

Check and add recursive
: You may check if there are somewhere in working copy unversioned items. After selecting **Subversion** → **Working copy** → **Check for unversioned items** a dialog appears where all not versioned items are listed. Hitting **OK** adds all marked items to working copy, items you don't want versioned you should unmark before.

Drag and drop
: Mark in Konqueror or any other compatible file browser items you want to add and drag them to kdesvn. You may drop them on folders inside the opened working copy and then kdesvn copy the dropped items to it and add the items.

2.2.4.2 Deleting items from working copy and unversion

Deleting items is always recursive. For example, when you delete a folder all subitems will be deleted, too. Mark what you want and select **Subversion** → **General** → **Delete selected files/dirs** menu item. Items will be unversioned and removed from disk.

2.2.5 Displaying logs

The log display may start with **Ctrl**+**L** when one or none item is marked in overview. Depending on your settings (see Table 5.2) kdesvn retrieves some log entries beginning on 'HEAD' and displays them.

2.2.5.1 The log display dialog

On the left side you see the list of log entries retrieved by kdesvn. The log message is stripped-down to one line to fit. The full log message appears in the upper-right window when marking an entry. In the lower-right window you will see the list of changed files. These windows *only* appear if this list is fetched, depending on your settings.

With the buttons on the top you may select another range of logs to display.

> IMPORTANT
> This range is called without any pre-limit, so be careful when using this feature on large repositories.

The buttons on the bottom of the dialog always work on the item selected for log, not on the item selected in the list of changed files. So when you select **Diff previous** there, it makes the diff over all changed items in this revision if they are equivalent or below the subversion entry selected before for retrieving logs. Same for **Diff revisions**.

Annotate of course works only if item to log is a file.

On all list entries (both lists) you have context menu enabled for some extra operations. In left list this is **Set version as left side of diff** (i.e. beginning revision) and **Set version as right side of diff** (i.e. end or target of diff, in unified the part marked with a +++). If you selected these revisions them are marked with some small arrows.

2.3 Working on repositories

Simple repository browsing may be done within Konqueror, Dolphin or similar file browsers: Open a URL with protocol described in Section 2.1 (the variants starting with 'k') and them will display the content. So simple operations like copy, move and delete may work. When appending a query '?rev=*xxx*' the listing comes from a specific revision. Format of revision-query is described in appendix A, some more information about KIO::ksvn are in chapter 3

All work except **Copy** may only done when browsing at revision HEAD.

2.3.1 Restoring deleted items

In subversion restoring deleted items is a copy operation of item at specific revision. So when you plan restoring view repository at revision before item was deleted. Select **Subversion** → **Repository** → **Select browse revision** and enter the wanted revision. Now kdesvn displays the content at this term. Mark the entry you want to restore, select **Subversion** → **General** → **Copy** (**Ctrl+C**). Inside the following dialog the target is always at HEAD revision, the source is at revision you selected for browsing. Fill in the path and click **OK** and copy starts. After successful copy switch browsing back to HEAD revision and the restored item should appear.

2.3.2 Importing folders

Due restrictions of subversion itself only folders, not single files, may imported.

2.3.2.1 With drag and drop

Mark in any compatible file manager the folder you want to import and drag it to the folder entry in kdesvn where you want to import it.

2.3.2.2 Select folder to import with directory-browser

Mark the folder where you want to import a new folder. Then select **Subversion** → **General** → **Import folders into current** and select your wanted folder.

2.4 Other Operations

2.4.1 Merge

Open repository or working copy, mark item you want to merge and select **Subversion** → **General** → **Merge**. Enter in the following dialog the values wanted. If opened from repository, source 1 and source 2 are filled, when open from within working copy, target is filled with current selected item. The handling of this parameter is a little bit different between using the internal diff of subversion or an external merge-program like KDiff3. The target must ALWAYS a local folder or file. You may switch between external or internal merge with the check box **Use external merge**.

2.4.1.1 Internal merge

The meaning is exactly like from within subversion's own command line tool. When source1 and source2 are equal, start and endrevision must different. If sources aren't equal, start-revision is assigned to source1 and end-revision to source2. The target MUST be a working copy otherwise subversion will send an error message.

The check boxes have following meanings:

Recursive
: Do all operations recursive when working with folders.

Handle unrelated as related items
: If set, unrelated items will be diffed as if they were related. Otherwise subversion will remove one side and add it on other side again.

Force delete on modified/unversioned items
: If not set and merge would require deletion of an modified or unversioned item the subversion merge fails. Otherwise this items will be deleted.

Just dry run without modifications
: If set, subversion sends only notifications what it would do but doesn't modify the working copy.

2.4.1.2 Using external program for merge

See Section 5.3 for details how to setup the external merge tool. kdesvn generates the command line as described there. Before it does following:

1. Assign start-revision as revision to source 1 and end-revision to source 2. Then checks if them are different (path and/or revision). If yes, it is a three-way merge otherwise a simple merge from source to target. If source 2 is left empty it is a simple merge, too.
2. Make an export into temporary folder. If it is a simple merge only source 1 at start-revision, otherwise both sources with their revisions. If item is a file and not a folder then get the content at a specific revision.
3. Generate the call to your external merge program as you setup in Settings. The error output will displayed in log window so you may see what is going wrong (if something is gone wrong).

In difference to internal merge target may a flat folder/file not under version control because external tools do not care about it.

If recursive is not set, the export is done as a flat export. Be care: when doing this with working copies externals will *not* be exported.

2.4.2 Resolving conflicts

kdesvn itself hasn't a module for resolving conflicts but you may use external software from within kdesvn. In Section 5.3.4 is a description how to setup up this application.

When marking an item with status set to 'conflicted' (you'll see a red cross in listview on such items) you may select **Subversion** → **Working copy** → **Resolve conflict**. or from within context menu **Resolve conflict** (only on conflicted items) kdesvn now starts the program you setup (or the default one). After finish this job you should mark item as resolved (**Subversion** → **Working copy** → **Mark resolved**) otherwise you will not able to commit your changes.

2.5 Properties used by kdesvn for configuration

2.5.1 Bugtracker integration

The TortoiseSVN project designed a nice property system for integrating bugtracker into subversion GUI. The current version of kdesvn does not support extra fields in commit box (will follow later) and doesn't understand all but following properties:

Property	Description	Example
`bugtraq:url`	Holds the URL to bugtracker. It has to contain the `%BUGID%` marker.	https://bugs.kde.org/show_bug.cgi?id=%BUGID%

`bugtraq:logregex`	Contains one or two regular expressions, separated by a newline. If only one expression is set, then the bare bug ID's must be matched in the groups of the regex string. If two expressions are set, then the first expression is used to find a string which relates to the bug ID but may contain more than just the bug ID (e.g. 'Issue #123' or 'resolves issue 123'). The second expression is then used to extract the bare bug ID from the string extracted with the first expression. Please keep care about not wanted spaces after the regular expression and don't forget the brackets around the number description.	Single (usable with TRAC):
		`#(\d+)`
		Now all numbers like *#190* will parsed and translated to an URL in log output.
		Two expressions:
		`[Ii]ssue #?(\d+)(,? ?#(\d+))* (\d+)`
		REMEMBER
		Keep care of white spaces after the `(\d+)`! This is one of the most common errors causing these expressions do not match!

Table 2.1: Bugtracker Integration Properties

On local opened repositories (i.e. `file://` protocol) and on working copies these properties will searched upward from opened folder until found or the subversion top level is reached. On repositories opened via network (all except `file://` protocol) it is only searched on opened folder itself.
Support for multiple sets of these properties may follow later, (e.g., in subfolder extra values for other tracker etc.) but in most cases evaluating single tracker links should be enough.

2.6 The revision tree

The revision tree try to display the whole history of an item so user may get a better feeling about the history of an item. It has to scan the complete log of the repository 'cause it requires some information subversion does not (and I think will not) give. Because this produces a lot of traffic the revision tree uses always the internal log cache.

2.6.1 Requirements

The revision tree is generated via dot. So for a working revision tree the graphviz package must installed.

2.7 Internal log cache

kdesvn may use an internal log cache for some operations. Mostly it is used for the revision tree but when viewing simplified log in offline mode, too. The caches are organized as SQLite databases stored in `$HOME/.sqvnqt/logcache`. Every numbered file is a storage for a different repository.

> WARNING
> The databases may get large! So you may disable automatic update of log cache in settings.

You may simply remove a database, than no cached log will be given back for that repository (and no revision tree!) but when you do not disable automatic filling the cache on next open this repository or an associated working copy of it the cache will be refilled again.

2.7.1 Offline mode

kdesvn may work without network access, i.e. you can switch off the network access for kdesvn. This may be useful when working without network like on notebooks. In such cases kdesvn always gets the logs from the internal cache. This log is reduced to base functions due to technical reasons, so the cached log may (but not must) differ from real log. Differ means that it will not display all copy operations even **Log follows node changes** is set.

2.7.2 Log cache and revision tree

The revision tree will only use the log cache because otherwise it must get always the logs again. It will *not* refresh the log cache (but this may changed in later releases).

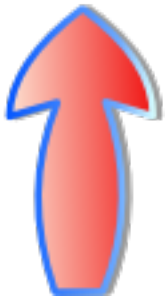
2.8 Meaning of icon overlays

Entries may be marked with overlayed icons when not in 'normal' state.

This entry is locked. In last column the owner of lock is listed. You may setup if kdesvn should check for locked items in repository, too. But be careful: depending of the kind of the server this may take a long time!

This entry must be locked before edit and commit. Until you'll not set a lock subversion keeps it read-only.

This entry or - if a folder - an entry below has newer version in repository.



This entry or - if a folder - an entry below was changed on local disc.

This entry is locally added to subversion but not yet committed.

This entry is locally deleted from subversion but hasn't yet been committed.

This entry (or entry below if folder) got conflicts to resolve with last update.

2.9 kdesvn and passwords

kdesvn/subversion is able to save passwords. Saving passwords is always a security risk, but may let a graphical frontend more usable.

2.9.1 Not saving passwords

Most secure way, but sometimes unhandy with GUIs like kdesvn. In particular the background processes of kdesvn would always ask for a password in case the repository has restricted access for reading operations like update and status. The same for 'commit' and so on. So if you are not saving passwords you should disable **Start check for updates when open a working copy** and so on.

2.9.2 Saving passwords in KWallet

Secured password storage used by a lot of KDE programs like KMail and Konqueror. If you are saving passwords and mostly using kdesvn you should use this. Keep care that the encrypted storage isn't a high-secure storage. Details see KWallet documentation.

2.9.3 Saving to subversion's own password storage

This is not recommended 'cause the passwords are stored as clear text! Not believing? Take a look into the files in `~/.subversion/auth/svn.simple`. You should only use this if you are frequently using other clients than kdesvn like rapidsvn or esvn or the original svn command line client. If you're using the command line client mostly for checkouts or updates which do not require a password and kdesvn for commit/move/copy you should use KWallet instead.

2.9.4 Internal password cache

You may activate an internal password cache which will hold passwords as long kdesvn is running in memory. So you must not enter a password twice even if you didn't save it in wallet.

2.9.5 Special case svn+ssh

When using subversion via svn+ssh password storage may done via ssh and ssh-agent. For this you must have ssh access to the remote machine and repository. When want to store something you should use the public key authentication of ssh, not the password authentication. (In fact ssh prefer the public key authentication). For this you must put your public ssh-key on the target, e.g., the repository system. SSH passwords will never be handled by subversion passwords storage or KWallet or internal password cache.

If you don't want asked for the password of your SSH key you can use the ssh-agent, with selecting the menu **Subversion** → **Add ssh identities to ssh-agent** you may store your SSH-key password for your current session so no further entering of your password is needed.

Chapter 3

Konqueror, KIO, kdesvn

3.1 Description

As of version 0.7.0 of kdesvn it comes with some modules integrating some commands directly into Konqueror menus.

KIO protocols

Implements handlers for the following protocols:

- ksvn+http
- ksvn+https
- ksvn+file
- ksvn+ssh
- ksvn

These protocols are designed only for repositories, not for working copies. For example, `ksvn+file:// / path` must point to the beginning of a repository different to the application itself or KPart. Working copies may browsed with Konqueror.

For browsing at a specific revision you may append the query '?rev=*revision*' to the URL.

Context menus

kdesvn installs context menus for Konqueror. Them may be seen with right mouse click in the browser window (only in standard view, not any KPart) so it is possible to do most used actions direct from within Konqueror (or any other file managers that read Konqueror context menus like Dolphin). This is done via a call to the command line variant of kdesvn.

3.1.1 Usage of KIO outside Konqueror - an example

Every application which uses the KIO library may use these protocols. So it would be possible to retrieve all differences between two revisions with KDiff3 without any deep knowledge.

Example 3.1 Retrieving differences between revisions using KDiff3 and KIO::ksvn

```
kdiff3 \
  ksvn://anonsvn.kde.org/home/kde/trunk/KDE/arts?rev=423127 \
  ksvn://anonsvn.kde.org/home/kde/trunk/KDE/arts?rev=455064
```

Let KDiff3 print all differences between two revisions.

> NOTE
> Using this within kdesvn (diff'ing two revisions) is MUCH faster because the internal mechanisms of subversion are used.

3.2 Programmers information about KIO::ksvn

> NOTE
> You may skip this if not interested in KIO programming.

KIO::ksvn::special knows the following operations.

3.2.1 Command list

Command name	Numeric id	Parameter list	Implemented in the current version?
Checkout	1	KURL repository, KURL target, int revnumber, QString revkind The target will *not* be modified, e.g., but the content will be checked out without creating a subfolder! For example, the source may be `http://server/repos/project/trunk`, target `/home/user/proj/` then the contents of trunk will copied into `/home/user/proj/` not `/home/user/proj/trunk/` !	Yes

Update	2	KURL url, int revnum, QString revstring If revnum < 0 the revstring is parsed. Format of revstring is described in Appendix.	Yes
Commit	3	KURL::List urls urls is a list of local urls to commit. Will ask for log message.	Yes
Log	4	int startrevnumber, QString startrevstring, int endrevnumber, QString endrevstring, KURL::List Use this with care - this may produce a lot of data.	Yes
Import	5	KURL targetrepository, KURL sourcepath	Yes
Add	6	KURL	Yes
Del	7	KURL::List	Yes
Revert	8	KURL::List Revert in KIO is always non-recursive, no questions (calling app must do it itself)	Yes
Status	9	KURL item, bool checkRepos, bool recurse item - the item whos info should be checked, checkRepos - check if there are newer versions in the repository, recurse - whether to check recursive or not.	Yes
Mkdir	10	KURL::List	Yes
Resolve	11	KURL, bool recursive	Yes

Switch	12	KURL working_copy_path, KURL new_repository_url, bool recursive,int revnumber, QString revkind	Yes
Diff	13	URL uri1, KURL uri2, int r1, QString rstring1, int r2, QString rstring 2, bool recursive For difference between repository file:/// and working copy setup working copy urls without a protocol!	Yes

Table 3.1: Command overview for KIO::ksvn::special

3.2.2 Return values

Return values may be given via metadata, see apidoc for details.

Key	**Possible value**
path	Path of the item action was made on, e.g. given url
action	Numeric action type
kind	kind of item (mostly folder or file)
mime_t	Subversion mimetype of item
content	State of content (subversion value)
prop	State of properties (subversion value)
rev	Resulting revision or revision worked on
string	Internal defined human readable message.
loggedaction	Subversion defined action string on item (A,M,D)
loggedcopyfrompath	If copied from which path? (may be empty)
loggedcopyfromrevision	If copied at which revision? (may be < 0)
loggedpath	On which single path the entries logged action and so on are set. (path is set to the calling url)
diffresult	a line of difference output

Table 3.2: Content of metadata

Chapter 4

Using kdesvn via command line

4.1 Overview

Some subversion operations may be used via the command line, e.g., like a standard command line client but user interaction is made through the UI. The standard syntax is **`kdesvn exec command parameter url`**.

If a single revision on a single URL is wanted it may be set as a parameter of the URL

```
svn://your-server/path-to-repository/item?rev=<your-rev>
```

This will overwrite the option `-r <rev>`.

A revision may be given as a number or one of `HEAD` or `BASE` or as date format like {YYYY-MM-DD}.

4.2 Command list

If in following overview as possible parameter `-r revision` is given, this revision may be set as **`url?rev=the-revision`**.

Command	Meaning	Accepted options
commit (or ci)	commit changes of item to repository	
log	Print log of item	-r startrevision:endrevision -l limit_display
cat	Display content of item	-r revision
copy (or cp)	Copy item inside working copy or repository. If target isn't given, kdesvn will prompt.	
move (or mv, rename)	Move/Rename item inside working copy or repository. If target isn't given, kdesvn will prompt.	
get	Get content of item and save it	-r revision -o <outputfile> (output is required!)
blame (or annotate)	annotate file	-r startrevision:endrevision

update	Update item in working copy	-r revision
diff	Diff two revisions of item or diff two items at specific revision	-r startrev:endrev
info	Detailed information about the item	-r revision
checkout (or co)	Checkout repository-path into a new working copy path. Target path and source revision will be asked for.	
checkoutto (or coto)	Checkout repository-path into a new working copy path. The difference of the source path and source revision will be asked for.	
export	Export repository- or working copy-path into directory. Target path and source revision will be asked for.	
exportto	Export repository- or working copy-path into directory. Source path and source revision will be asked for.	
delete (del, remove, rm)	delete url(s) from repository or working copy.	
add	add URL to working copy. URL must belong to a working copy (its not an import!)	
revert (or undo)	undone current changes to working copy. May only used on working copy urls!	
checknew (or addnew)	check in given URL for new, unversioned items and add them to working copy if wanted.	
tree	displays revision tree of item (only first argument), if URL with '?rev=*xxx*' given, this revision is the peg-revision.	-r startrev:endrev
lock	lock url(s), if `-f` is given then existing locks are broken.	-f
unlock	unlock url(s), if `-f` is given then not owned locks are broken or non existing locks are ignored.	-f
help	displays this page	

Table 4.1: Subversion commands

Parameter	Possible values	allowed for
`-r`	*`revision`* or *`startrev:endrev`*	all except commit
`-R`	(none)	all except commit
`-o`	*`filename`*	get
`-l`	*`number`*	log
`-f`	(none)	(un-)lock

Table 4.2: Parameter for subversion commands

4.2.1 The 'log' command

Log command displays a dialog containing the log of the given URL. With subversion 1.2 or above it accepts a limit, i.e. how many entries it has to display.

Inside that dialog you may select log entries and get the differences between them.

Example 4.1 Display the last 20 commit logs

```
kdesvn exec log -l 20 -r HEAD:1 myfile.c
```

Beware of order of revision: You want go from HEAD to START for the LAST one. So you must give revision HEAD as starting point, otherwise you would get the first 20 entries.

4.2.2 The 'diff' command

You get differences between revisions of an item or between two items inside same working copy or repository. When diff'ing revisions of an item, revisions may be given as `-r STARTREV:ENDREV`. When diffing an item inside a working copy without any revisions it prints the diff against repository.

Example 4.2 Print difference against repository, i.e. local changes

```
kdesvn exec diff myfile.c
```

Example 4.3 Print difference between revisions

```
kdesvn exec diff -r 21:20 myfile.c
```

When diffing two items revisions may be appended to URL of items. e.g.:

```
http://server/path/item?rev=HEAD
```

Example 4.4 Diffing two tagged versions

```
kdesvn exec diff http://www.alwins-world.de/repos/kdesvn/tags/rel_0_6_2  ↩
    http://www.alwins-world.de/repos/kdesvn/tags/rel_0_6_3
```

Chapter 5

Settings

Settings can be changed from the setup dialog. They are separated into some subdialogs.

5.1 General

Size of listviewicons	How big (square) the icons in main list view should be
Show file info	If a small tooltip should appear when moving mouse over an item
Mark item status with icon overlay	When an item isn't in normal subversion state it may get an overlay. (See Section 2.8)
Items sorting order is case sensitive	If the sort order in main window is case sensitive or not, i.e. if 'a' not equal to 'A'.
Display ignored files	Show items marked in subversion for ignore or not.
Maximum log messages in history	How many log messages kdesvn should remember.
Display colored annotate	Use colors defined in the settings dialog for annotations.
Use navigation panel	Toggle the display of the repository tree view.

5.2 Subversion and timed jobs settings

Start check for updates when open a working copy	When open a working copy start a check for updates in background
Start fill log cache on open	If set kdesvn start (re-)filling the log cache when open a repository or working copy if repository URL is not assigned local (`file://`)

Check if items has 'svn:needs-lock' set	Check on working copies if an item has this property set and if yes, displays a special overlayed icon. Setting this may let listings get real slow.
Get file details while remote listing	When checked, kdesvn get more detailed info about file items when making a listing to remote repositories. So you may see remote locks in overview. Be careful: This may let listings REAL slow!
Gain item info recursive	When activated, **info** on folders will gain information about all items below, may be real slow.
Store passwords for remote connections	Storing passwords is often a security problem. Subversion stores its passwords into a flat file, i.e. passwords in the plain text! So be careful setting this flag and see next entry, too. This item only says if saving passwords is the default, you may change it for specific realms inside login dialog.
Store passwords into KDE Wallet	When saving passwords, the plain text file from subversion is a security hole. kdesvn is able saving them into encrypted KDE wallet instead there (starting with version 0.12.0) and use them. On other hand other subversion clients aren't able reading them so you must enter them if using tools like svn command line tool or rapidsvn, too. As long as subversion does not encrypt password storage you should think twice about it.
Use internal password cache	When a password is not stored persistent kdesvn may hold it until kdesvn ends, so you may not need to enter it again for each operation. This cache is never persistent, i.e. it will not be saved anywhere.
Log follows node changes	If checked log follows copy operations.
Log always reads list of changed files	The log command may read list of changed files in commit. This is useful and in most cases it cost not real more traffic.
Review items before commit	When doing a commit kdesvn may make a check what to do. For example, if unversioned items are below which may be added, list items changed and current operation will commit. This cost a depending on tree more or less time.
Hide new items in commit box	Should unversioned items displayed in commit dialog or not.
Update modifies instead of conflicts	If set, a local addition at the same path as an incoming addition of the same node kind results in a normal node with a possible local modification, instead of a tree conflict.

Update makes missing parents	If set, create any non-existent parent directories also by checking them out at depth=empty

Table 5.2: Subversion

Check modified items every N seconds Check for updated items every N seconds	If set check for updates or modified items in working copy when network is enabled on regular base

Table 5.3: Timed jobs

5.3 Diff & Merge

Diff ignores content type

Only interesting when diff are made with subversion itself. When set, than subversion diff ignores the content type of entries. Otherwise it will not output diffs from binaries.

Diff in revision tree is recursive

When set, diffs made from within revision tree view are made recursive like in all other cases, too. Otherwise only changes belonging to that folder items are shown. How that is made depends on how you let diffs generate (from subversion itself or from external viewers).

Diff ignores white space changes

Ignore changes in the amount of white space (option `-b` to diff)

Diff ignores all white spaces

Ignore all white space (option `-w` to diff)

Prefer external merge program

Set if merge with external program is preferred and not Subversion's merge

Use Git diff format

Show copies as add

5.3.1 Use external diff display

Selects an external application to display diffs. Default is Kompare.

5.3.2 External diff display

Defines what kdesvn is using for external display of differences and how it will be called. There are three ways:

<program> <parameter>

The difference will generated with subversion and put direct into standard input of the external program (i.e., no temporary files needed)

<program> <parameter> %f

The difference will be generated with subversion, saved into a temporary file and the parameter `%f` will be replaced with that file name. This may be used, for instance, with a simple call to less or text viewer.

<program> <parameter> %1 %2

kdesvn let the external program make the difference. %1 and %2 will be replaced with required values (file names or folder names). kdesvn stores content to compare in a temporary environment (when folders does an 'export', when single file, does a 'cat') when needed and cleans up after closing external program or closing itself.

Prefer external merge program

Select if in merge dialog **Use external merge** should be checked or not as default.

5.3.3 External merge program

Setup the program and options for using when subversion's builtin merge isn't wanted. The default is **kdiff3 %s1 %s2 %t**. The order of substitution variables isn't important, and them may there more than once, e.g. like `kdiff3 -o %t %s1 %s2 %t`. This stuff is only tested with meld and KDiff3. Think about that external programs mostly do not know anything about subversion **ignore** parameter so them may show lot more than expected.

5.3.3.1 Variable substitutions for external merge program

%s1

Substituted with source number one.

%s2

Substituted with source number two. If this is empty or this is equal to source one and start and end revision is same, this variable will skipped. So be careful setting up command lines like **xxdiff --title1 %s1 --title2 %s2 %s1 %s2 %t**.

%t

Substituted with target.

5.3.4 Conflict resolver program

You may use an external program like KDiff3 for resolving conflicts, the default is **kdiff3 %o %m %n -o %t**.

5.3.4.1 Variable substitution for external conflict resolver

In parenthesis after each description is an example how subversion would call the files. These options are designed for KDiff3, because at this moment this is the only one application supporting all parameters required for a good conflict resolving.

%o or %l

Old (local, left) version. This means the lower revision number, i.e. the start point of conflicted changes. (`foo.cc.r2`)

%m or %w

Mine (working) version of file, i.e. what you had changed against old version. (`foo.cc.mine`)

%n or %r

New (remote, right) version of file. For example, version someone other had made. (`foo.cc.r3`)

%t

Target name, e.g. the origin name. For KDiff3 (as an example) this would be name after the `-o` parameter (= output file). (`foo.cc`)

5.4 KIO / command line

Show log after executing a command

Should a dialog open with the log of last subversion command when it was executed via command line or Konqueror action menu.

Minimum log lines to show

If **Show log...** is set, what is the minimum of lines before such a dialog will shown. So you may set that such window will only shown when interesting output was generated (commit log and so on)

Do not display context menu in Konqueror

If set, no action menu entry for kdesvn is made in Konqueror.

Do not display entries in toplevel action menu

If set, kdesvn will not show some extra actions inside **Action** menu of Konqueror or Dolphin.

KIO operations use standard log message

When making operations on a repository via the kdesvn KIO protocol from within Konqueror (i.e., 'ksvn+...' protocols) on large operations like moving or copy folders kdesvn would ask for a log message for each item. This is a behavior of Konqueror. When this option is set, the KIO implementation from kdesvn will not ask for a log message but set a standard log message. This flag is not for the operations from kdesvn action menu for Konqueror but only copy/move/mkdir/delete made with Konqueror or other file managers on a KIO-url.

Standard message

The message kdesvn KIO should set on operations from within Konqueror when the flag above is set. Default is **Revision made with kdesvn KIO**.

KIO can overwrite

If this flag is set, you will have a simple write support for existing items. e.g. you can open files in your editor and save them direct without checking out them before (kdesvn will do it in background).

Use this only if you are sure what you are doing.

KIO shows progress messages

If set KIO shows in KDE's Plasma detailed information about current operation. Error messages of KIO will always displayed and can *not* be switched off.

Chapter 6

Command Reference

6.1 The main kdesvn window

6.1.1 The File Menu

File → Open (Ctrl+O)

Open a local working copy or a repository previously checked out

File → Recent opened urls

This is a shortcut to open recently opened repositories. Clicking on this item opens a list to the side of the menu with several of the most recently opened local working copies or a repositories. Clicking on a specific item will open it in kdesvn

File → New (Ctrl+N)

Open a window with a new instance of kdesvn

File → Subversion Admin

Menu items with administration tasks for subversion repositories like:

Create and open new repository
Dump repository to file
Hotcopy a repository
Load dump into repository

For more information please read the output of **svnadmin --help**

File → Close (Ctrl+W)

Close current opened repository or working copy

File → Quit (Ctrl+Q)

Quits kdesvn

6.1.2 The Bookmark Menu

See Konqueror help.

6.1.3 The Subversion Menu

GENERAL SUBVERSION ACTIONS

Subversion → General → History of item (Ctrl+L)

Display lifetime log of the currently selected item. Be careful, this list may be really big!

Subversion → General → History of item ignoring copies... (Ctrl+Shift+L)

Displays the history log of selected item without following copies.

Subversion → General → Details (Ctrl+I)

Displays detailed information about selected item(s)

Subversion → General → Blame

Makes an annotated list over all checkins. That may consume time!

Subversion → General → Blame range

Annotate a range of commits for a file.

Subversion → General → Cat head

Shows the content of the last committed version of that entry. (May be different to working copy version if working on a WC!)

Subversion → General → Move (F2)

Move or rename item inside working copy or in repository

Subversion → General → Copy (Ctrl+C)

Copy item inside working copy or in repository

Subversion → General → Delete selected files/dirs (Del)

Delete selected entries. If working in a working copy you must commit your deletions afterwards.

Subversion → General → New folder

Create a new folder.

Subversion → General → Import folders into current

Select folders you want to import into the current selected directory

Subversion → General → Checkout a repository

Creates a new working copy of a repository

Subversion → General → Export a repository

Exports a repository to file system, i.e. creates a clean revision tree without subversion information.

Subversion → General → Lock current items

Mark current items as locked. Read the subversion handbook before using this!

Subversion → General → Unlock current items

Remove locks on current items. Read the subversion handbook before using this!

WORKING COPY

Subversion → Working copy → Update to head

Update working copy to HEAD of repository

Subversion → Working copy → Update to revision...

Update working copy to a specific revision of repository

Subversion → Working copy → Commit (Ctrl+#)

Commit changes inside working copy for selected items to repository.

Subversion → Working copy → Diff local changes (Ctrl+D)

Display local changes as diff-output (without network access). This is the difference only to last state the working copy was updated to, not against the version in repository.

Subversion → Working copy → Diff against HEAD (Ctrl+H)

Diff's current working copy against head of repository.

Subversion → Working copy → Properties (Ctrl+P)

View/Edit properties assigned with current entry.

Subversion → Working copy → Add selected files/dirs (Insert)

Add selected files and/or folders to version control.

Subversion → Working copy → Revert current changes

Revert changes made in working copy and updates to last updated state.

Subversion → Working copy → Mark resolved

Mark conflicted items as not conflicted and remove associated files.

Subversion → Working copy → Merge two revisions

Merge two revisions of entries into working copy.

Subversion → Working copy → Ignore/Unignore current item

Edit property of parent folder of the current item so that selected item will be marked as ignored if not set, otherwise remove it from ignore list.

Subversion → Working copy → Cleanup

Clean up the working copy and remove (commit-)locks if any

Subversion → Working copy → Switch repository

Switch the root of the current working copy.

REPOSITORY

Subversion → Repository → Checkout current repository path

Create a working copy from the current selected entry if a directory.

Subversion → Repository → Export current repository path

Create a clean copy on local file system from the current selected entry if a directory.

VIEW ACTIONS

Subversion → View → Refresh view (F5)

Refresh the current status of all displayed items. This will list each item asked for to be displayed at current status.

Subversion → View → Unfold File Tree / Fold File Tree

Expand or collapse the repository tree view.

Subversion → Log Cache → Stop updating the log cache

The log is cached and used to display the revision tree of a repository. Depending on the Internet connection, size of the subversion repository and history length this may take much time. Therefore you can stop updating the cache using this action.

Subversion → Add ssh identities to ssh-agent

Store your SSH-key password for your current session so no further entering of your password is needed.

6.1.4 The Database Menu

Database → Show database content

Displays an overview about the cache database content for known repositories and allows to delete the cache or repository and access to repository settings.

Database → Settings for current repository

Display a dialog to adapt the settings for the cache, logs and statistics.

6.1.5 The Settings and Help Menu

Apart from the common KDE **Settings** and **Help** menu items described in the the Settings Menu and Help Menu of the KDE Fundamentals kdesvn this additional menu entry:

Settings → Quick settings

Load last opened URL on start
Reload last opened URL if no one is given on command line.

Logs follow node changes

Display ignored files
Show items marked in subversion for ignore or not.

Display unknown files
Show files not added to the subversion repository.

Hide unchanged files
Use this action to show only changed files and provide a clear and concise overview.

Work online
Working offline the log cache will not be updated, therefore the cached log may differ from real log.

Chapter 7

Credits and License and Thanks

Many thanks to contributors:

- Andreas Richter ar@oszine.de - for qt4 port of svnqt
- Michael Biebl biebl@teco.edu - for a lot of help, ideas, implementation and hints.
- Bram Schoenmakers bramschoenmakers@kde.nl - for KDE-specific hints, dutch translation, cleaning up my code.

And thanks to all other translators (I never thought that I get that fast such a lot of translations!) and for all of your positive and negative feedback. It helped a lot.

If you want your own translation for kdesvn and may want help please read on the KDE Localization page or ask on kde-i18n-doc@kde.org mailing list.

Appendix A

Syntax for revisions

Revisions may be given in the same form as for the standard svn client. That means: number, keyword or date.

Number

A number greater or equal -1. -1 means 'unspecified revision', 0 is the beginning. Normally, these numbers should not be used (most operations will fail with that).

Keyword

One of

- HEAD
- BASE
- COMMITED
- PREV
- START
- WORKING

The keywords are case sensitive! For example, head is not the same as HEAD.

Date

Date in form {YYYY-MM-DD}. It must real MM or DD - e.g. 2005-1-1 must be written as {2005-01-01}.

Appending a specific revision to an URL will always made via '?rev=*revision*'.