

The **svn-prov** package

Use SVN Id keywords for package, class and file header

Martin Scharrer
martin@scharrer-online.de

Version v3.1862 - April 24, 2010

1 Introduction

This package is directed to authors of L^AT_EX packages and classes which use the version control software Subversion¹ (SVN) for their source files. It introduces three macros which are Subversion variants of the standard L^AT_EX header macros **nProvidesPackage**, **nProvidesClass** and **nProvidesFile** which are used to identify package, class and other files, respectively. Instead of providing the package/class/file name and date manually they are extracted from a Subversion Id keywords string which is updated automatically by every time the source file is committed to the repository.

A similar package exists for RCS, the pre-predecessor of Subversion, in the pgf² bundle which is called **pgfrcs**. For further support for Subversion keywords see the author's other package **svn-multi**³.

2 Usage

The following macros need an Id keyword which can initially be written as '**\$Id:\$**' and will be expanded by Subversion into the following format at the next commit:

```
$Id: h lenamei hrevisioni hdatei hetimei hauthori $
```

e.g. for the source file of this document:

```
$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $
```

For this to work the Subversion **property svn:keywords** must be set to (at least) '**Id**' for the source file(s). e.g. using the command line:

```
svn propset 'svn:keyword' 'Id' h lename(s)i
```

More information about using Subversion in the L^AT_EX workflow can be found in the PracT_EX Journal issue 2007-3⁴.

¹WWW: <http://subversion.tigris.org/>

²CTAN: <http://tug.ctan.org/pkg/pgf>

³CTAN: <http://tug.ctan.org/pkg/svn-multi>

⁴URL: <http://www.tug.org/pracjourn/2007-3/{skiadas-svn|ziegenhagen|scharrer}>

```

\ProvidesPackageSVN[h le namei]{$Id: $}[hversion and/or description i][hdescription i]
\ProvidesClassSVN[h le namei]{$Id: $}[hversion and/or description i][hdescription i]
\ProvidesFileSVN[h le namei]{$Id: $}[hversion and/or description i][hdescription i]

```

All of these macros await a valid Subversion Id keyword string as a mandatory argument. The file name and date is extracted from this string. For cases when the file source is not stored in the correct file but packed inside a different one, like a `.dtx` file, the correct file name can be provided by an optional argument. Because the file extension of package and class files is predefined and therefore ignored this is not needed for them when they are packed inside a corresponding `.dtx` file, i.e. one with the same base name.

As with the standard macros mentioned above additional information can be given optionally. Since v0.3 the SVN macro provide two optional arguments (before only one). If only one optional argument is given it is taken as a description text which may start with an potential version number. This version number must start with ‘v’ and not include spaces and is extracted from the description. Alternatively the version number and the description can be provided using two separate optional arguments. If no optional argument is given the default string `nrevinfo` (see below) is used instead.

All three optional arguments can include the following macros which are only valid inside them, but not afterwards⁵:

`nrev` File revision.

`nRev` File revision followed by a space.

`nrevinfo` The default information used: “(SVN Rev: `hrevision i`)”.

`nfilebase` File base name (file name without extension).

`nfileext` File extension (without leading dot).

`nfilename` File name.

`nfiledate` File date (in the format YYYY/MM/DD).

`nfilerev` File revision, like `nrev`.

```

\GetFileInfoSVN{hnamei}
\GetFileInfoSVN*

```

This macro sets the macros `nfilebase`, `nfileext`, `nfilename`, `nfiledate`, `nfileversion`, `nfilerev`, `nfileinfo` and `nfiletoday` to the corresponding values of the file given by `hnamei`. The file must have been read/loaded before and use both a `nProvides...SVN` macro and `nDefineFileInfoSVN`, otherwise

Non-star
version added
in v3.
2010/04/11

⁵They can be set using `nGetFileInfoSVN`

the above macros will be set to `\relax`. The `hnamei` can be either the real filename or the optional short name used with `nDefineFileInfoSVN`.

The star version of this macro provides the file information of the last file which called one of the `nProvides...SVN` macros.

The macros `nfileversion` and `nfileinfo` hold the file version and description taken from optional argument of the `nProvide...SVN` macro. The version is defined only if this argument starts with ‘v’ and is otherwise empty. It includes all text up to the first space. The `nfiletoday` macro generates a text representation of the `nfiledate` using the `ntoday` macro. The format can be adjusted to a different language with the `\datehlanguagi` macro from the `babel`⁶ package. The other macros are described above.

`\DefineFileInfoSVN[hnamei]`

Defined a set of macros which provide the information collected by a previous `nProvides...` macro. The macros have the form `\hnamei@hdatai` where `hnamei` is by default the filename either with the file extension (general files) or without (packages and classes). This default can be overwritten by the optional argument. The `hdatai` stands for **version**, **rev** (revision), **date** and **info** (the information part without the version number) and, since v3, file name **base** and **ext**(ension) as well as **today**, which prints the **date** in the format of `ntoday`.
Example: Applied to the `.dtx` file of this very package the following macros are defined:

New in v1.
2009/05/03

Updated in v3.
2010/04/24

Macro	Definition
<code>nsvn-prov.dtx@version</code>	v3.1862
<code>nsvn-prov.dtx@rev</code>	1862
<code>nsvn-prov.dtx@date</code>	2010/04/24
<code>nsvn-prov.dtx@info</code>	DTX for svn-prov.sty
<code>nsvn-prov.dtx@base</code>	svn-prov
<code>nsvn-prov.dtx@ext</code>	dtx
<code>nsvn-prov.dtx@today</code>	April 24, 2010

The style file however would get macros like `nsvn-prov@version`. Because ‘-’ is not a letter the macros can only be accessed using `\csname`. Therefore the optional argument `[svnprov]` is used to name the macros `nsvnprov@version` etc..

3 Examples

The following examples illustrate the usage of the provided macros and how they call the equivalent standard macros internally. The example **results** are produced by expanding the corresponding example **code** while the standard

⁶CTAN: <http://tug.ctan.org/pkg/babel>

provide macros are locally redefined to typeset their own name and arguments in verbatim style. This does not only simplifies the generation of this document but makes this examples also test cases which allow the package author to test the result of the defined macros.

While mostly the package macro is used here the usage is identical to the class and file macros. Of course before this macros are used it must be made sure that the `svn-prov` package is loaded which is done by using the following code direct before them:

```
\RequirePackage{svn-prov}
```

Minimal usage

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
```

is equivalent to:

```
nProvidesPackage{svn-prov}[2010/04/24 (SVN Rev: 1862)]
```

The following code:

```
\ProvidesClassSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
```

is equivalent to:

```
nProvidesClass{svn-prov}[2010/04/24 (SVN Rev: 1862)]
```

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
```

is equivalent to:

```
nProvidesFile{svn-prov.dtx}[2010/04/24 (SVN Rev: 1862)]
```

Normal Usage

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesPackage{svn-prov}[2010/04/24 v1.0 Example Description]
```

The following code:

```
\ProvidesClassSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesClass{svn-prov}[2010/04/24 v1.0 Example Description]
```

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesFile{svn-prov.dtx}[2010/04/24 v1.0 Example Description]
```

Normal Usage with only Description

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [Example Description]
```

is equivalent to:

```
nProvidesFile{svn-prov.dtx}[2010/04/24 Example Description]
```

Normal Usage with separate Version and Description

The following code:

```
\ProvidesFileSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0][Example Description]
```

is equivalent to:

```
nProvidesFile{svn-prov.dtx}[2010/04/24 v1.0 Example Description]
```

Overwriting Name

The following code:

```
\ProvidesPackageSVN[othername]
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesPackage{othername}[2010/04/24 v1.0 Example Description]
```

Overwriting Name including unneeded Extension

The following code:

```
\ProvidesPackageSVN[othername.sty]
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesPackage{othername}[2010/04/24 v1.0 Example Description]
```

Overwriting Name using Macros

The following code:

```
\ProvidesFileSVN[\filebase.cfg]
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.0 Example Description]
```

is equivalent to:

```
nProvidesFile{svn-prov.cfg}[2010/04/24 v1.0 Example Description]
```

Using Macros in File Information String

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.\Rev Example Description]
```

is equivalent to:

```
nProvidesPackage{svn-prov}[2010/04/24 v1.1862 Example Description]
```

Adding Text to Default Information

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.\Rev Extra Text \revinfo]
```

is equivalent to:

```
nProvidesPackage{svn-prov}[2010/04/24 v1.1862 Extra Text (SVN Rev:
1862)]
```

Getting the File Information

The following code:

```
\ProvidesPackageSVN
  {$Id: svn-prov.dtx 1862 2010-04-24 14:19:07Z martin $}
  [v1.\Rev Extra Text \revinfo]
\GetFileInfoSVN*
% ...
\begin{tabular}{l@{\ : \ }l}
  File Name      & \filename      & \\
  File Base Name & \filebase      & \\
  File Extension & \fileext       & \\
  File Date      & \filedate      & \\
  File Revision  & \filerev       & \\
  File Version   & \fileversion   & \\
  File Info      & \fileinfo      & \\
\end{tabular}
```

results in:

```
File Name      : svn-prov.dtx
File Base Name : svn-prov
File Extension  : dtx
File Date      : 2010/04/24
File Revision   : 1862
File Version    : v1.1862
File Info       : Extra Text (SVN Rev: 1862)
```

The correct package file extension ‘.sty’ for `nfileext` can be forced by using `[\filebase.sty]` as a first optional argument.

4 Implementation

```
15 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
```

`\ProvidesClassSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
17 \def\ProvidesClassSVN{%
18   \svnprov@generic\ProvidesClass{\svnprov@filebase@}%
19 }
```

`\ProvidesFileSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
21 \def\ProvidesFileSVN{%
22   \svnprov@generic\ProvidesFile{\svnprov@filebase@.\.
      svnprov@fileext@}%
23 }
```

`\ProvidesPackageSVN`

Calls the generic macro with the original LaTeX macro and the string to be used as filename.

```
25 \def\ProvidesPackageSVN{%
26   \svnprov@generic\ProvidesPackage{\svnprov@filebase@.
      }%
27 }
```

`\svnprov@generic`

Stores the arguments (1: original macro, 2: file mask (full filename if only base is used?)). Then tests if a explicit file name was given as optional argument. If not the file name from the SVN Id string is used.

```
29 \def\svnprov@generic#1#2{%
30   \def\svnprov@ltxprov{#1}%
31   \def\svnprov@filemask{#2}%
32   \begingroup
33   \svnprov@catcodes
34   \@ifnextchar{[%
35     {\svnprov@getid}%
36     {\svnprov@getid[\svnprov@svnfilename]]}%
37 }
```

`\svnprov@catcodes`

Sets the normal catcodes for all characters required by the `getid` macro.

```
39 \def\svnprov@catcodes{%
40   \catcode'\ =10%
41   \catcode'\$=3%
42   \@makeother\:%
43   \@makeother\-%
44 }
```

Enforce normal catcodes for the definition of the `Id` scanning macros. This makes sure that all scan patterns have the same catcodes during definition and execution.

```
46 \begingroup
47 \svnprov@catcodes
```

`\svnprov@getid`

Saves first argument as filename and calls the scan macro with the second. A fall-back string is provided to avoid `TeX` parsing errors.

```
49 \gdef\svnprov@getid[#1]#2{%
50   \endgroup
51   \def\svnprov@filename{#1}%
52   \svnprov@scanid #2\relax $%
53   Id: unknown.xxx 0 0000-00-00 00:00:00Z user $\.
54   empty\svnprov@endmarker
55 }
```

`\svnprov@scanid`

Parses the `Id` string and tests if it is correct (`#1=empty`, `#8=nrelax`). If correct the values are stored in macros and the next macro is called. Otherwise a warning message is printed. In both cases any remaining text of the parsing procedure is gobbled before the next step.

```
56 \gdef\svnprov@scanid#1$%
57   Id: #2 #3 #4-#5-#6 #7 $#8{%
58   \def\next{%
59     \begingroup
60     \PackageWarning{svn-prov}{Invalid SVN Id line .
61       found! File name might be
62       '#2' or '\expandafter\strip@prefix\meaning\
63       @filef@und'. This occurred}}{}{}{}%
64   \endgroup
65   \svnprov@gobbleopt
```

```

64 }%
65 \ifx\relax#1\relax
66   \ifx\relax#8\empty
67     \def\svnprov@svnfilename{#2}%
68     \svnprov@splitfilename{#2}%
69     \def\svnprov@filerev@{#3}%
70     \def\svnprov@filedate@{#4/#5/#6}%
71     \def\svnprov@filetoday@{\svnprov@@today.
72       {#4}{#5}{#6}}%
73     \def\next{\begingroup\svnprov@catcodes\
74       svnprov@buildstring}%
75   \fi
76 \fi
77 \expandafter\next\svnprov@gobblerest
78 }% $

```

End of area with enforced catcodes.

```

78 \endgroup

```

`\svnprov@@today`

Prints `ntoday` with the given date.

```

80 \def\svnprov@@today#1#2#3{%
81   {\year#1\month#2\day#3\relax\today}%
82 }

```

`\svnprov@splitfilename`

Expands the argument and initialises the file base macro before it calls the next macro with the expanded argument and a dot to protect for `TEX` parsing errors. The `nrelax` is used as end marker.

```

84 \def\svnprov@splitfilename#1{%
85   \edef\g@tempa{#1}%
86   \let\svnprov@filebase@\@gobble
87   \expandafter
88   \svnprov@splitfilename@\g@tempa.\relax
89 }

```

`\svnprov@splitfilename@`

The second argument is tested if it is empty (end of file name reached). If not empty the first argument is concatenated to the file base macro and the macro calls itself on the second argument. This ensures correct handling of file name which contain multiple dots.

If the second argument was empty it is tested if the file base name is still in its initialised state which means that there is no file extension. Then the file base is defined to the first argument and the extension as empty. Otherwise the file extension is defined to the first argument and the file base macro is unchanged because it is already correct.

```

91 \def\svnprov@splitfilename@#1.#2\relax{%
92   \if&#2&
93     \ifx\svnprov@filebase@\@gobble
94       \gdef\svnprov@filebase@{#1}%
95       \gdef\svnprov@fileext@{}%
96     \else
97       \gdef\svnprov@fileext@{#1}%
98     \fi
99     \let\next\relax
100  \else
101    \xdef\svnprov@filebase@{\svnprov@filebase@.#1}%
102    \def\next{\svnprov@splitfilename@#2\relax}%
103  \fi
104  \next
105 }
```

`\svnprov@gobblertest`

Simply gobbles everything up to the next endmarker.

```

107 \def\svnprov@gobblertest#1\svnprov@endmarker{}
```

`\svnprov@endmarker`

This is the end marker which should never be expanded. However it gets defined and set to an unique definition which will gobble itself if ever expanded.

```

109 \def\svnprov@endmarker{\@gobble{svn-prov endmarker}}
```

`\svnprov@gobbleopt`

Gobbles an optional argument if present.

```

111 \newcommand*\svnprov@gobbleopt[1][{}]
```

`\svnprov@defaultdesc`

Default description text to be used. Does not include the file date which is prepended later.

```

113 \def\svnprov@defaultdesc{%
114   (SVN Rev:\space\svnprov@filerev@)%
115 }

```

\svnprov@buildstring

First aliases the internal macro to user-friendly names and then builds the info string. Finally the stored original LaTeX macro is called with the filename and information.

```

117 \newcommand*\svnprov@buildstring[1][\ .
      svnprov@defaultdesc]{%
118   \@ifnextchar{[]%
119     {\svnprov@buildstring@{#1}}%
120     {\svnprov@buildstring@{#1}[\relax]]}%
121 }
122 \def\svnprov@buildstring@#1[#2]{%
123   \endgroup
124   \begingroup
125     \let\rev\svnprov@filerev@
126     \let\filerev\svnprov@filerev@
127     \def\Rev{\rev\space}%
128     \let\revinfo\svnprov@defaultdesc
129     \let\filebase\svnprov@filebase@
130     \let\fileext\svnprov@fileext@
131     \ifx\fileversion\@undefined
132       \def\fileversion{v0.0}%
133     \fi
134     \edef\filename{\filebase.\fileext}%
135     \xdef\svnprov@filename{\svnprov@filename}%
136     \ifx\svnprov@filename\filename\else
137       \svnprov@splitfilename{\svnprov@filename}%
138     \fi
139     \let\filename\svnprov@filename
140     \ifx\relax#2\empty
141       \xdef\svnprov@fileinfo@{#1}%
142       \svnprov@getversion{#1}%
143       \global\let\svnprov@filedesc@\svnprov@filedesc@
144       \global\let\svnprov@fileinfo@\svnprov@fileinfo@
145     \else
146       \xdef\svnprov@fileversion@{#1}%
147       \xdef\svnprov@filedesc@{#2}%
148       \xdef\svnprov@fileinfo@{#1 #2}%
149     \fi
150   \endgroup
151   \svnprov@ltxprov{\svnprov@filemask}%

```

```

152     [\svnprov@filedate@
153     \ifx\svnprov@fileinfo@\empty\else
154         \space
155         \svnprov@fileinfo@
156     \fi
157 ]%
158 }

```

\GetFileInfoSVN

The macro provides the file information of the given file, or (the star version) the last file which called one of the above `nProvides...` macros. For this the internal macros are simply copied to user-friendly names.

This macro is inspired by the macro `nGetFileInfo{h le namei}` from the `doc` package.

```

160 \def\GetFileInfoSVN#1{%
161     \ifx*#1\relax
162         \let\filebase\svnprov@filebase@
163         \let\fileext\svnprov@fileext@
164         \let\filename\svnprov@filename
165         \let\filedate\svnprov@filedate@
166         \let\filerev\svnprov@filerev@
167         \let\fileversion\svnprov@fileversion@
168         \let\fileinfo\svnprov@filedesc@
169         \let\filetoday\svnprov@filetoday@
170     \else

```

Given argument could be filename or short name. If a short name exists for the argument it was a filename is is defined as such, otherwise the filename is read from the `\hshort namei@long` macro.

```

172         \expandafter\let\expandafter\@gtempa\csname#1.
173             @short\endcsname%
174         \ifx\@gtempa\relax
175             \def\@gtempa{#1}%
176             \expandafter\let\expandafter\filename\csname#1.
177                 @long\endcsname
178         \else
179             \edef\filename{#1}%
180         \fi
181     \expandafter\let\expandafter\filebase\csname\
182         @gtempa @base\endcsname
183     \expandafter\let\expandafter\fileext \csname\
184         @gtempa @ext\endcsname
185     \expandafter\let\expandafter\filedate\csname\
186         @gtempa @date\endcsname

```

```

182     \expandafter\let\expandafter\filerev \csname\
        @gtempa @rev\endcsname
183     \expandafter\let\expandafter\fileversion\csname\
        @gtempa @version\endcsname
184     \expandafter\let\expandafter\fileinfo\csname\
        @gtempa @info\endcsname
185     \expandafter\let\expandafter\filetoday\csname\
        @gtempa @today\endcsname
186 \fi
187 }

```

\DefineFileInfoSVN

Defines macros in the form \h **lename**i@hxxxi, where hxxxi is date, version, rev(ision), info, (file name)base and ext(ension).

```

189 \newcommand*\DefineFileInfoSVN[1][\svnprov@filemask]{.
    %
190     \expandafter
191     \edef\csname\svnprov@filemask @short\endcsname{#1}%
192     \expandafter
193     \edef\csname#1@long\endcsname{\svnprov@filemask}%
194     \expandafter
195     \let\csname#1@base\endcsname\svnprov@filebase@
196     \expandafter
197     \let\csname#1@ext\endcsname\svnprov@fileext@
198     \expandafter
199     \let\csname#1@date\endcsname\svnprov@filedate@
200     \expandafter
201     \let\csname#1@version\endcsname\
        svnprov@fileversion@
202     \expandafter
203     \let\csname#1@rev\endcsname\svnprov@filerev@
204     \expandafter
205     \let\csname#1@info\endcsname\svnprov@filedesc@
206     \expandafter
207     \let\csname#1@today\endcsname\svnprov@filetoday@
208 }

```

\svnprov@getversion

Checks if the argument (a file description) starts with ‘v’. If so everything until the first space is taken as version number. Otherwise the whole text is taken as description without version. Special care is taken to avoid a parser error if there is no space included.

```

210 \def\svnprov@getversion#1{%
211   \edef\@tempa{#1\space}%
212   \expandafter\svnprov@@getversion\@tempa\
        svnprov@endmarker
213 }
214 \def\svnprov@@getversion{%
215   \@ifnextchar{v}%
216     {\svnprov@getversion@}%
217     {\svnprov@getversion@@}%
218 }
219 \def\svnprov@getversion@#1 #2\svnprov@endmarker{%
220   \gdef\svnprov@fileversion@{#1}%
221   \ifx&#2&%
222     \gdef\svnprov@filedesc@{}%
223   \else
224     \xdef\svnprov@filedesc@{\svnprov@zapspace#2\
        svnprov@endmarker}%
225   \fi
226 }
227 \def\svnprov@getversion@@#1 \svnprov@endmarker{%
228   \gdef\svnprov@fileversion@{}%
229   \gdef\svnprov@filedesc@{#1}%
230 }
231 \def\svnprov@zapspace#1 \svnprov@endmarker{#1}

    Finally, call the macros for this package itself.
233 \ProvidesPackageSVN{$Id: svn-prov.dtx 1862 2010-04-24 .
        14:19:07Z martin $}%
234 [\svnprov@version\space Package Date/Version from .
        SVN Keywords]
235 \DefineFileInfoSVN[svnprov]

```