pure-systems pure::variants Connector for Version Control Systems User Manual

Contents

1 pure-systems pure::variants Connector for Version Control Systems
2 Product Information
3 Synopsis
4 Concept
5 Installing the Connector
6 Using the Connector
7 Implementing a Wrapper
8 Documents / Resources
9 Related Posts

pure-systems pure::variants Connector for Version Control Systems

Product Information

The pure::variants Connector for Version Control Systems (VCS) is a software extension developed by pure-systems GmbH. It is designed to synchronize local files used in variant transformations with files from a Software Configuration Management (SCM) repository. This connector provides a generic transformation module that utilizes a VCS-specific wrapper program to connect pure::variants with the VCS client.

It is important to note that there are special connectors available for popular SCM systems such as CVS and Subversion, known as the pure::variants Connector for CVS and the pure::variants Connector for Subversion. This user manual explains how to use the connector and provides instructions on implementing the interface of the VCS-specific program. It also includes an example of a wrapper for the CVS VCS. A printable version of this document is available.

Software Requirements

- The Connector for Version Control Systems (VCS) is an extension for pure::variants and is compatible with all supported platforms.

Synopsis

- The pure::variants Connector for Version Control Systems (VCS) is used to synchronize local files used in variant transformations with files from a Software Configuration Management repository.
- This extension provides a generic transformation module using a VCS specific wrapper program to connect pure::variants with the VCS client.

Note

- For the popular SCM systems CVS and Subversion special connectors are available, i.e. the pure::variants
- Connector for CVS and the pure::variants Connector for Subversion.
- This document describes how the connector is used and how the interface of the VCS specific program has to

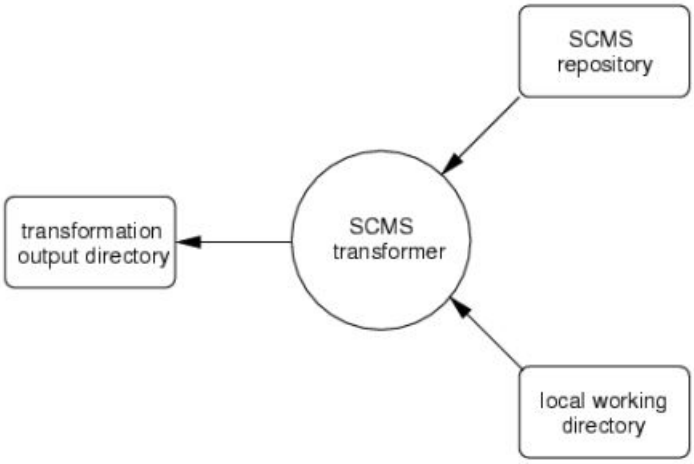
be implemented. Additionally it provides an example of a wrapper for the popular CVS VCS.
- A printable version of this document is available.

Software Requirements

- The Connector for Version Control Systems (VCS) is an extension for pure::variants and is available on all supported platforms.

Concept

Figure 1. VCS Synchronization



The purpose of this connector is to synchronize files in the transformation output directory with corresponding files from a local or remote VCS repository in the following way. VCS-managed files are modeled using ps: simfile source elements. If a such modeled file does not yet exist in the transformation output directory it is checked out from the VCS repository. If a local VCS repository (local working directory) is given, the file is first searched there. Otherwise the file is checked out from the remote VCS repository.

If a file already exists in the transformation output directory it is synchronized with the corresponding file in the VCS repository according to its revision and change state. Again it is first searched in the local working directory if given. To synchronize files with various SCM systems, the connector uses an VCS specific wrapper program for the actual communication with the VCS. The implementer of this wrapper program is responsible for implementing the above algorithm. It is within the discretion of the implementer what functionality is provided by the wrapper and what by the VCS client. If for instance a specific VCS does not support local working directories, the implementer of the wrapper program could rebuild this or a similar functionality in the wrapper.

Note: Please ask the implementer of the wrapper program if some functionality is missing or what other functionality is implemented by the wrapper and the specific VCS.

Client Authorization

The connector relies on external authorization of the VCS client. If during the synchronization process the VCS client program requests a username and/or password interactively, it is therefore either not possible to use the VCS connector or the wrapper program has to provide a functionality to pass the required authorization information to the VCS client (if the client itself does not provide a corresponding functionality).

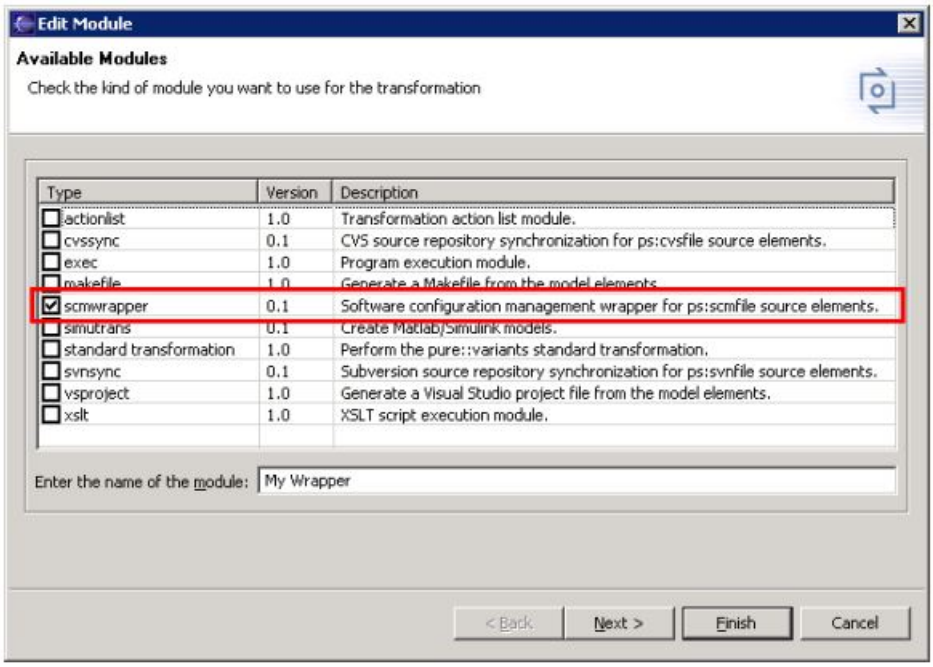
Note: Please ask the implementer of the wrapper program or consult the documentation of the VCS to get information about how to provide authorization information for the VCS client if required.

Installing the Connector

- Please consult section “pure::variants Connectors” in the pure::variants setup Guide for detailed information on how to install the connector (menu Help -> Help Contents and then pure::variants Setup Guide -> pure::variants Connectors).

Using the Connector

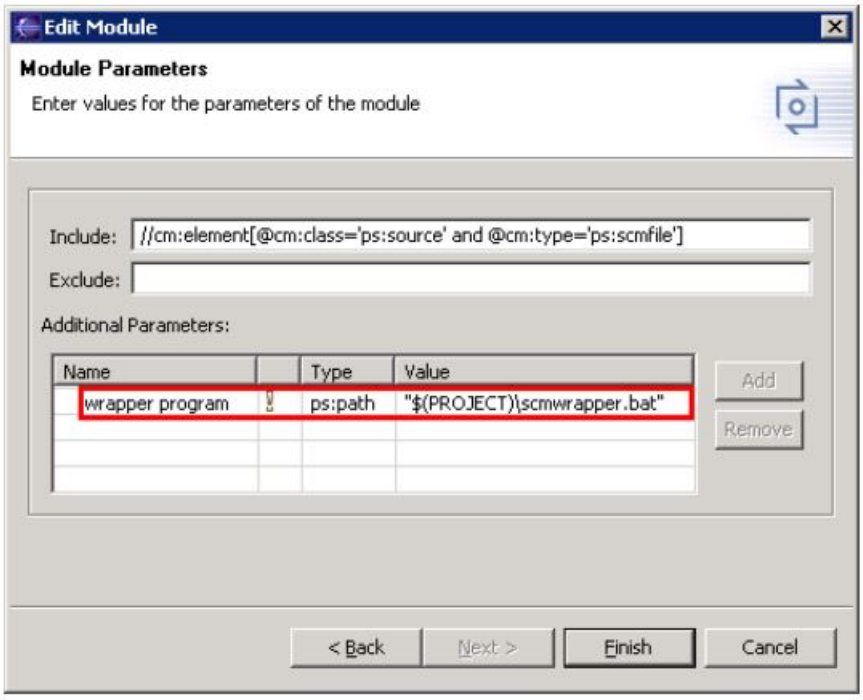
Figure 2. Available Transformation Modules



After installing the connector a new transformation module is available called “scmwrapper”. To see if the connector is available open the properties of a configuration space by choosing Properties from the context menu of the configuration space. On the Configuration Space page change to the Transformation Configuration tab and then click on button Add. This opens a dialog showing the list of available transformation modules. It has to contain the “scmwrapper” module as shown in Figure 2, “Available Transformation Modules”.

Connector Configuration

Figure 3. VCS Connector Configuration

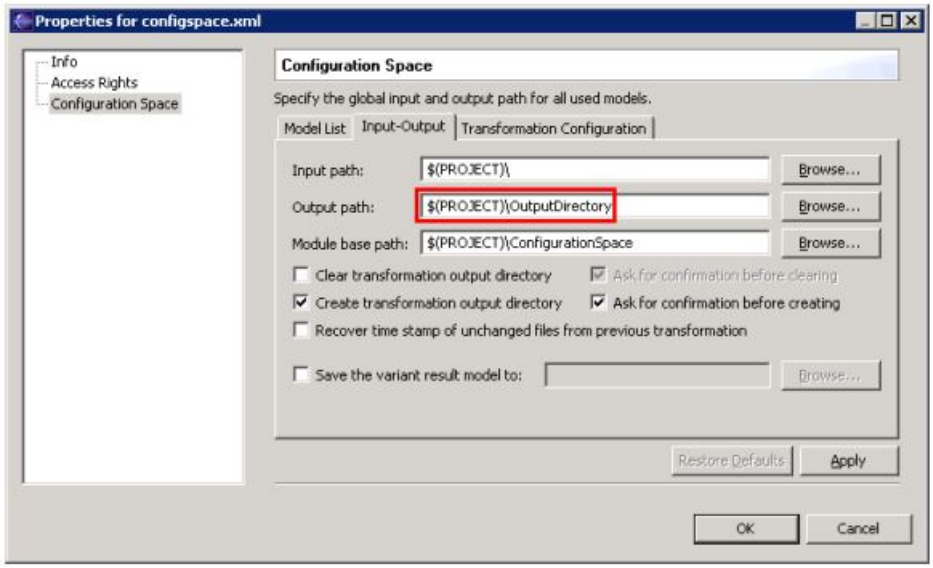


As described above the VCS connector uses a wrapper program for the actual file synchronization. The command line to invoke this wrapper program has to be entered on the configuration page of the connector. Open the list of available transformation modules as described in the beginning of this chapter. Ensure that the check box of the module “scmwrapper” is checked. Then click on button Next to open the configuration page of the VCS connector (see Figure 3, “VCS Connector Configuration”).

The “scmwrapper” module has exactly one parameter.

Wrapper Program: The command line for the wrapper program call. The wrapper program is executed for each ps:scmfile source element in the input models. The information about the current ps:scmfile source element is passed to the wrapper using environment variables.

Figure 4. Transformation Output Directory



For a valid transformation configuration a transformation output directory has to be specified. Into this directory the files from the VCS repository are written. The output directory is specified on the Input-Output page of the configuration space properties dialog (see Figure 4, “Transformation Output Directory”). For more information on how to setup a transformation see section “Transformation Configuration Page” in chapter “5.3.7. Configuration

Space Editor” of the pure::variants Eclipse Plugin User’s Guide.

ps:scmfile Source Element

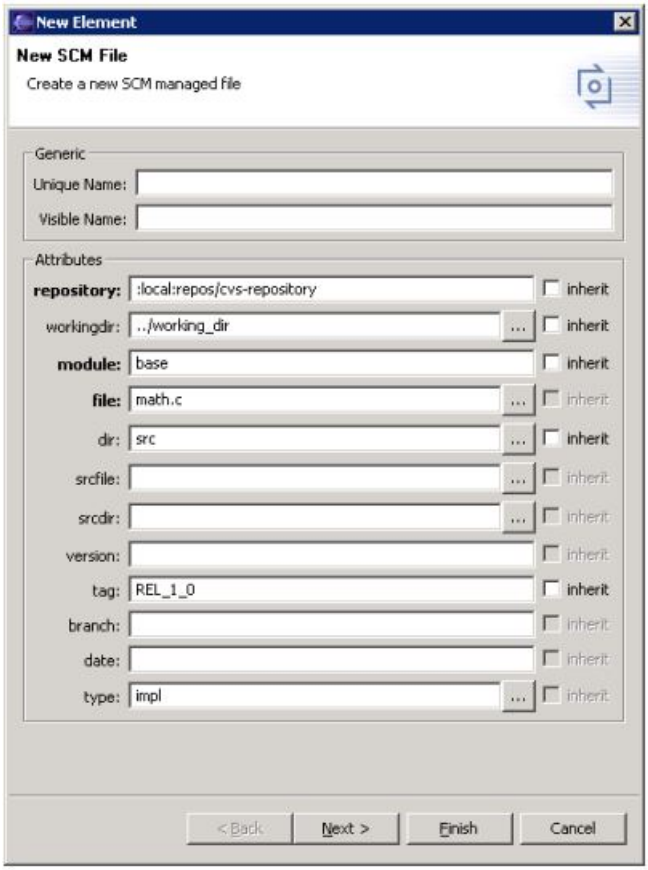
VCS managed files are modeled using ps:scmfile source elements. The attributes of this source element are listed below. These attributes represent a common set of the information needed by an VCS to synchronize a file. For specific SCM systems some of the attributes may have no meaning or may have a slightly different meaning as for other SCM systems. Thus only the attributes for specifying the VCS repository location and the name and path of the file to synchronize are mandatory.

Note: Please consult the documentation of the VCS or ask the implementer of the wrapper program for the VCS specific meaning of these attributes.

- **repository:** [Mandatory] The VCS repository containing the file.
- **dir:** [Mandatory] The directory part of the VCS managed file relative to the given VCS module.
- **file:** [Mandatory] The file name part of the VCS managed file.
- **srcdir:** [Optional] The directory part of the VCS managed file relative to the given module. To be used if the source directory location differs from the target directory location.
- **srcfile:** [Optional] The file part of the VCS managed file. To be used if the source file name differs from the target file name.
- **type:** [Optional] The type of the file according to attribute “type” of the ps:file source element.
- **module:** [Optional] The name of the module in the VCS repository containing the file.
- **branch:** [Optional] The name of the VCS repository branch containing the file.
- **tag:** [Optional] The tag of the VCS managed file.
- **version:** [Optional] The version of the VCS managed file.
- **date:** [Optional] The date specifying the version of the VCS managed file.
- **workingdir:** [Optional] The local working directory containing the local copies of the files managed in the VCS repository.

To create a new ps:scmfile source element choose New->SCM File from the context menu of a part element. This opens the New SCM File wizard as shown in Figure 5, “New SCM File Wizard”.

Figure 5. New SCM File Wizard



After setting up the transformation and the input models the VCS synchronization of the modeled files is started by performing a variant transformation. A complete example of the use of the pure::variants Transformer for Software Configuration Management is presented by the “SCM Module” variant management example project. This example project can be installed in the current workspace by choosing New->Example from the context menu of the Variant Projects view and then Examples->Variant Management->SCM Module.

Implementing a Wrapper

The wrapper program is responsible for realizing the synchronization of a file with an VCS repository (local and/ or remote). During the variant transformation the wrapper is called for every ps:scmfile source element of the input models. The information about the current ps:scmfile source element, i.e. the values of its attributes, and the transformation input and output directories are passed to the wrapper using environment variables. With this information the wrapper can construct a suitable VCS client call to synchronize the file it was called for.

The following tasks have to be performed by the wrapper

1. Prepare the transformation output directory. For instance if it does not exist, checking out files to it may fail depending on whether the VCS client expects the target directory to be present or not.
2. Calculate the location of the source file in the VCS repository and the location of the target file in the transformation output directory. If necessary also calculate the location of the source file in the local working directory.
3. Call the specific VCS client.
    - If needed the wrapper also has to interact with the user or has to implement functionality not provided by the VCS client. For instance if the VCS client program does not write the target file by itself, the wrapper has to save the checked out file to the calculated target file location. Or a VCS could not support local working directories. In this case the wrapper program could provide the missing functionality if desired.

But this depends on the wrapped VCS client and may strongly differ for different SCM systems.

Environment Variables

The following environment variables are set before the VCS wrapper program is executed.

- **SCM_INPUTDIR:** The path to the transformation input directory.
- **SCM_OUTPUTDIR:** The path to the transformation output directory.
- **SCM_MODULE:** The value of the ‘module’ attribute of the current ps:scmfile source element.
- **SCM_DIR:** The value of the ‘dir’ attribute of the current ps:scmfile source element.
- **SCM_FILE:** The value of the ‘file’ attribute of the current ps:scmfile source element.
- **SCM_SRCDIR:** The value of the ‘srcdir’ attribute of the current ps:scmfile source element.
- **SCM_SRCFILE:** The value of the ‘srcfile’ attribute of the current ps:scmfile source element.
- **SCM_BRANCH:** The value of the ‘branch’ attribute of the current ps:scmfile source element.
- **SCM_TAG:** The value of the ‘tag’ attribute of the current ps:scmfile source element.
- **SCM_VERSION:** The value of the ‘version’ attribute of the current ps:scmfile source element.
- **SCM_DATE:** The value of the ‘date’ attribute of the current ps:scmfile source element.
- **SCM_REPOSITORY:** The value of the ‘repository’ attribute of the current ps:scmfile source element.
- **SCM_WORKINGDIR:** The value of the ‘workingdir’ attribute of the current ps:scmfile source element.

Calculating File Locations

An essential task of the wrapper is to calculate the correct locations of the source and target files. The target files have to be written to the transformation output directory. The path to this directory is set in the variable SCM_OUTPUTDIR. The location of the file in the output directory is calculated by appending the name (variable SCM_FILE) and directory (variable SCM_DIR) of the file as follows:

- SCM_OUTPUTDIR + <path_separator> + SCM_DIR + <path_separator> + SCM_FILE

The path separator is for instance a backslash on Windows (e.g. dir\file.txt) and a slash on Linux (e.g. dir/ file.txt). For an VCS supporting modules (like CVS) it may be necessary to include the module name (variable SCM_MODULE) in the target path, e.g.

- SCM_OUTPUTDIR + <path_separator> + SCM_MODULE + <path_separator> + SCM_DIR + <path_separator> + SCM_FILE

The calculation of the source file location is slightly different from the calculation of the target file location since it may have to consider revision information. If the name of the source and target file is identical, and thus SCM_SRCFILE and SRC_SRCDIR are empty, the name of the source file is as follows.

- SCM_DIR + <path_separator> + SCM_FILE

If the name or location of the source and target file is different, SCM_SRCFILE resp. SCM_SRCDIR has to be used instead of SCM_FILE resp. SCM_DIR. E.g. if the directory part differs, then the source file location is as follows.

- SCM_SRCDIR + <path_separator> + SCM_FILE

If a local working directory is given, then the path to the working directory (variable SCM_WORKINGDIR) is prepended.

- SCM_WORKINGDIR + <path_separator> + SCM_DIR + <path_separator> + SCM_FILE

Depending on the VCS, revision and module information has to be added to the source file path. Please consult the documentation of the specific VCS to find out how files in the VCS repository are addressed.

Example Wrapper Program

The following batch file is an example wrapper for the CVS VCS. It supports revision information (date, version, tag, and branch) and modules. Local working directories are not supported by this CVS wrapper. This wrapper is part of the “SCM Module” example project. This example project can be installed in the current workspace by choosing New->Example from the context menu of the Variant Projects view and then Examples-> Variant Management->SCM Module.

For every ps:scmfile source element of the input models it performs a checkout operation on the given CVS repository for the file described by the ps:scmfile source element. For this purpose it uses the environment variables listed above to calculate the location and name of the source file in the CVS repository and the location and name of the target file in the local repository. Then it calls the CVS client with the calculated arguments and writes the checked out file to the target file location. First the variable client is set to the full path of the CVS client program that is used later in the script to perform the actual checkout operation.

- REM path to cvs client
- set client=”C:\Program Files\WinCvs 1.2\cvs.exe”

In the next step the revision information of the source file is collected. If a date is given, then this date is taken as the revision. Otherwise a given version, tag, or branch is taken, checked in this order. If no revision can be determined, the newest revision of the source file in the VCS repository will be checked out.

```
REM get file revision
set revision=
if not "%SCM_DATE%"=="" (
  set revision=-D %SCM_DATE%
) else (
  if not "%SCM_VERSION%"=="" (
    set revision=-r %SCM_VERSION%
  ) else (
    if not "%SCM_TAG%"=="" (
      set revision=-r %SCM_TAG%
    ) else (
      if not "%SCM_BRANCH%"=="" set revision=-r %SCM_BRANCH%
    )
  )
)
```

The next code blocks calculate the complete path to the source file as described above. The name of the source file is taken from the variable SCM_SRCFILE if not empty, otherwise from the variable SCM_FILE. The directory part is taken from the variable SCM_SRCDIR if not empty, otherwise from the variable SCM_DIR. Since CVS expects slashes as path separators, all backslashes in the path are replaced by slashes. Then the complete path is constructed using the file name, the directory part, and, if given, the module name taken from the variable SCM_MODULE.

```
REM get source file
set sourcefile=%SCM_FILE%
if not "%SCM_SRCFILE%"=="" set sourcefile=%SCM_SRCFILE%

REM get source file directory (replace backslashes with slashes, required by cvs)
set sourcedir=%SCM_DIR:\=/%
if not "%SCM_SRCDIR%"=="" set sourcedir=%SCM_SRCDIR:\=/%

REM remove trailing separator
if "%sourcedir:~-1%"=="/" set sourcedir=%sourcedir:~0,-1%

REM build complete source file path
if not "%sourcedir%"=="" set sourcefile=%sourcedir%/%sourcefile%
if not "%SCM_MODULE%"=="" set sourcefile=%SCM_MODULE%/%sourcefile%
```

After calculating the source file path the next code blocks calculate the complete path to the target file. The directory part of the target file path is taken from the variable SCM_DIR. Here slashes in the directory part are replaced by backslashes because the target file is located in the local file system using backslashes as path separators. As described above, the target file has to be written to the transformation output directory. Using the output directory (variable SCM_OUTPUTDIR), the directory part of the target file path, and the target file name (variable SCM_FILE), the full path to the target file is constructed.

```
REM get target file
set targetdir=%SCM_DIR:/=\%

REM remove trailing separator
if "%targetdir:~-1%"=="\" set targetdir=%targetdir:~0,-1%
set targetdir=%SCM_OUTPUTDIR%\%targetdir%

REM build complete target file path
set targetfile=%targetdir%\%SCM_FILE%
```

To be able to write the target file, it has to be ensured that the target directory exists. It is created if it doesn't exist.

- REM create the target directory
- if not exist “%targetdir%” mkdir “%targetdir%”

After calculating the source and target file locations and the revision of the source file to check out, the last step is to call the CVS client. The CVS client gets the repository location, the revision, and the source file path as input and writes the checked out file to stdout which is redirected to the target file location.

- REM call cvs client
- %client% -d “%SCM_REPOSITORY%” co -n -p %revision% “%sourcefile%” > “%targetfile%”

Documents / Resources

pure::variants - Connector for
Version Control Systems Manual

pure-systems GmbH

Version 3.2 (for pure::variants 3.2)

Copyright © 2003-2022 pure-systems GmbH

2022

Table of Contents

1. Synopsis

Note

1.1. Software Requirements

2. Concept

pure-systems pure::variants Connector for Version Control Systems [pdf] User Manual

pure variants, Connector for Version Control Systems, pure variants Connector for Version Control Systems, pure variants Connector, Connector