

Plastic SCM Administration

A guide for the Plastic SCM administrator

Plastic SCM 4

© 2006-2011 Codice Software

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Codice Software cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Table of Contents

1	Introduction	2
1.1	Plastic SCM	2
1.2	Components	2
2	Minimum requirements	4
2.1	Server	4
2.2	Client	4
3	Plastic SCM installation	6
3.1	Prerequisites	6
3.2	Server and Client installation	6
3.3	Server configuration	9
3.4	User authentication configuration	11
3.4.1	Authentication basics	11
3.4.2	Local users	11
3.4.3	Local users: name + ID	12
3.4.4	Active Directory Integrated Security	12
3.4.5	LDAP	12
3.4.6	User/Password	13
3.5	Server startup	19
3.5.1	Windows Systems	19
3.5.2	Linux Systems	20
3.6	Client configuration	21
3.6.1	Merge and Differences Tools configuration	23
4	Using a proxy server	26
4.1	Installing the Plastic SCM Proxy server	28
4.2	Configuring the clients	29
5	Creating and managing repositories	31
5.1	Creating a new repository	31
5.2	Listing available repositories	32
5.3	Archiving repositories	33
5.4	Reconnecting archived repositories	34
6	Database setup	36
6.1.1	Oracle specific options	39
7	Backup and restore	41
7.1	How to backup the embedded databases	41
7.2	Restore embedded databases	43
8	Archiving revisions	44
8.1	Why archiving revisions	44
8.2	How to archive my revisions	44
8.3	How are the archived revisions accessed	45
8.4	How to restore archived revisions	46

Figures

Figure 1. Installation directory.....	7
Figure 2. Component selection	7
Figure 3. Eclipse location	8
Figure 4. Copying files	8
Figure 5. Start system configuration.....	9
Figure 6. Server configuration welcome screen	9
Figure 7. Server configuration language selection.....	10
Figure 8. Server configuration authentication modes	10
Figure 9. LDAP Authentication configuration	10
Figure 10. Port configuration	11
Figure 11. User and password configuration screen	14
Figure 12. Login dialog	14
Figure 13. configuration files in user/password mode.....	15
Figure 14 umtoolgui usage.....	16
Figure 15. users.conf file format	18
Figure 16. groups.conf file format	18
Figure 17. server.conf contains <i>WorkingMode</i>	18
Figure 18. Managing Plastic SCM server service	20
Figure 19. Client configuration welcome screen.....	21
Figure 20. Client configuration language selection	21
Figure 21. Workspace server selection	22
Figure 22. Client authentication selection	22
Figure 23. Client AD LDAP authentication window.....	22
Figure 24. Example of proxy deployment	27
Figure 25. Example of network	27
Figure 26. Same network, improved by using two proxies	28
Figure 27. Proxy installer binaries folder	28
Figure 28. Cached data directory location.....	29
Figure 29: Proxy listening port	29
Figure 30: configuring a proxy server in the Plastic SCM client.....	30
Figure 31: creating a new repository in the Plastic SCM GUI client.....	32
Figure 32: New repository dialog	32
Figure 33. Repositories view.....	33
Figure 34: Database migration wizard	37
Figure 35: database migration: target database options.....	37
Figure 36: remember to stop the Plastic SCM server before migrating the database.	39
Figure 37: migration finished.....	39
Figure 38. Introduce the external data location path.....	46

About this guide

This guide describes the procedures associated with Plastic SCM installation and maintenance.

Audience

This guide is targeted to developers and system administrators, assuming familiarity with Plastic SCM and operating system concepts.

Online documentation

Besides this document and the rest of the guides, Plastic SCM provides online reference throughout its different client frontends.

On the command line interface, both Windows and Linux, this reference can be obtained with the command:

```
cm help
```

For extended information on a specific command, type:

```
cm help command
```

The graphical interface provides online reference through the Help menu.

Documentation errors

If you find any problem in this guide or any other part of the online reference, please report it using the following email address:

support@codicesoftware.com

1 Introduction

1.1 Plastic SCM

Plastic SCM is a Software Configuration Management system designed to handle software development teams of any size.

Plastic SCM provides high-end SCM capabilities without imposing any of the restrictions associated to these high-end systems like complex installation, operation and administration.

This guide assumes that the reader is familiar with the basic SCM concepts, with basic operating system usage through the command line and basic system administration.

The guide will show both system administrators and SCM managers how to install the SCM system, how to create repositories, workspaces and how to make backups.

1.2 Components

Plastic SCM uses a client / server architecture, divided into the following components:

- The server, responsible for storing all the project information, managing client access to that store.
- The Clients, run on the developers' machines, and are responsible for communicating user operations to the server. The supported clients are:
 - Command Line Interface (CLI): Provides access to Plastic SCM operations on the operating system shell. Very useful for task automation.

- Graphical User Interface (GUI): provides access to Plastic SCM operations using a graphic-oriented interface. It provides some graphical diagrams not available in the command line.
- Integrations with Integrated Development Environments such as:
 - Visual Studio integration: provides access to commonly used operations like checkout / checkin from within Visual Studio, as well as full access to GUI views like Branch Explorer, labels and checkouts.
 - Eclipse integration: provides access to the most used operations like checkout / checkin from within the Eclipse development environment.
 - IntelliJ integration: provides access to the commonly used operations from the IDEA IDE.
 - JDeveloper.
- Build Management tools integrations for the following products:
 - Cruise Control
 - Final Builder
- Integrations with task and issue management tools like:
 - Atlassian Jira
 - TechExcel DevTrack
 - RallyDev
 - Axosoft Ontime
 - Version One
 - Fog Creek's FogBugz
 - Trac
 - Bugzilla
 - Mantis

Visit our web page to get a full and up to date list of compatibilities at www.codicesoftware.com.

2 Minimum requirements

Below you will find the minimum hardware and software requirements needed to install and use Plastic SCM.

2.1 Server

The minimum system requirements to run the Plastic server are:

- Windows 2000 Server, Windows 2000 Professional, Windows 2003 Server or Windows XP SP2 Operating Systems. Server operating systems are strongly recommended.
- The following Linux distributions are supported: OpenSuSE, Fedora, Ubuntu, Debian, Gentoo, RedHat.
- Mac OS X: Tiger version or higher.
- .NET Framework 2.0 or higher.
- 512 MB RAM.
- Enough free hard disk space. Hard disk space will vary depending on the size of project's assets.

2.2 Client

Client machine requirements will vary depending on the Plastic SCM client in use. The GUI or IDE integrations will have higher requirements than the command line.

The supported operating systems are:

- Windows 2000 Professional, Windows XP SP2, Windows Vista, Windows 7
- Windows Server 2000 / 2003 / 2008
- Fedora Core 8 or higher
- Ubuntu 8.04 or higher
- RedHat RHEL 5
- OpenSuse 10.3 or higher, SLED 10 or higher
- CentOS 5
- Mac OS X 10.4 or higher

For windows, the .NET Framework 2.0 or higher is required. For Linux operating systems a mono package is distributed with the Plastic SCM installer.

The needed RAM will vary depending on:

- Command line client: is the smallest and needs few resources to work. Nevertheless, as a general rule, a 512 Mb RAM machine is strongly recommended.
- GUI client: 512 MB RAM
- Visual Studio Plug-in: same requirements as the IDE.
- Eclipse Plug-in: same requirements as the IDE.
- Other integrations: same requirements as the integrated package.

3 Plastic SCM installation

3.1 Prerequisites

The Plastic SCM server is installed as a Windows service or as a daemon in Linux / Mac. Administrative privileges are required for this step to succeed during installation of the server component.

3.2 Server and Client installation

Plastic SCM is distributed as a single installation package. Once started, the installer will let the user choose the components to install and performing the initial setup.

The table below details the different installer steps and options after selecting the installation process language and accepting the license agreement.

Choose the installation directory for the Plastic SCM components.

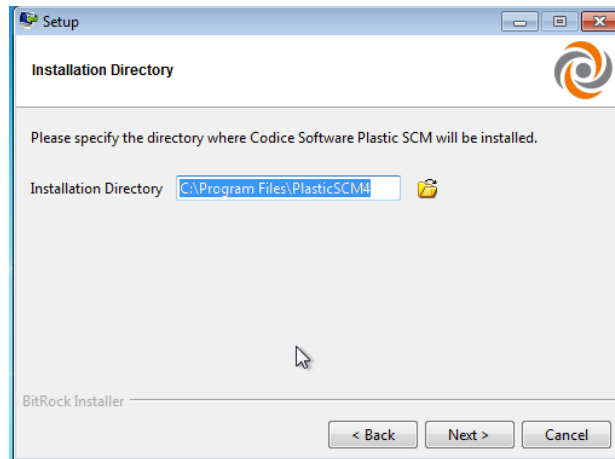


Figure 1. Installation directory

Component selection.
By default, the CLI and GUI clients are always installed.
Optionally, the Plastic SCM server and integrations with 3rd party tools can be installed. Below is a description of each component.

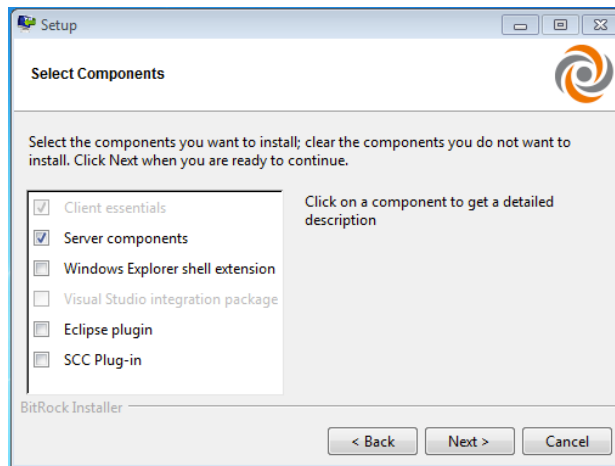


Figure 2. Component selection

The components available are:

- **Client essentials:** The command line client (the cm command) and the graphical user interface client (plastic.exe).
- **Server:** the Plastic SCM server
- **Windows Explorer shell extension:** installs the Shell Extension of Plastic SCM that integrates Plastic SCM with the Windows Explorer. The integration lets the user do most of the operations available on the Plastic SCM GUI client directly from the Windows Explorer.
- **Visual Studio integration package:** available for Visual Studio 2005, 2008 and 2010, this package offers most of the graphical tools found on the Plastic SCM GUI client right inside the Visual Studio environment.
- **Eclipse plugin:** the Plastic SCM integration with the Eclipse platform.
- **SCC plugin:** this is Plastic SCM implementation of MS-SCCI interface. It provides checkin/checkout/add and basic file-level operations.

The SCC plugin is used by many tools, especially in the Windows world, as a way to interface with the version control backend. Check the documentation of your tool for compatibility with the MS-SCCI

specification (also known as SCC). It used to be primarily used by Visual Studio, but deprecated since Visual Studio 2005. Plastic SCM offers a richer integration in the *Visual Studio integration package*.

If the Eclipse plugin is selected, the root folder for the Eclipse install needs to be selected, so that the installer can figure out where to copy the needed files.

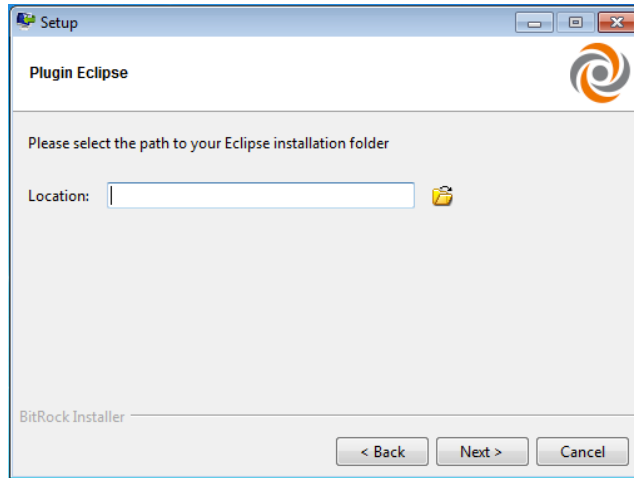


Figure 3. Eclipse location

Ready to install.

At this point the installer has all the needed information.

File copy progress.

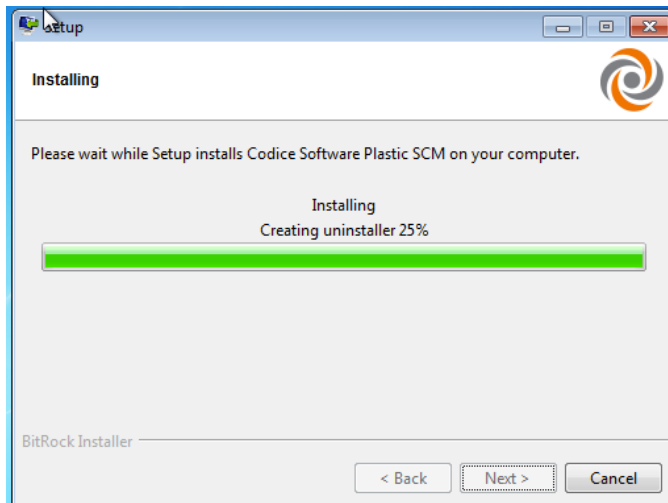


Figure 4. Copying files

Once the basic installation is complete, the installer asks if the server and client configuration wizards should be launched. If this is an upgrade (i.e. there was a previous version of Plastic and the installer just upgraded it), then this step can be skipped. Otherwise, the configuration needs to be finished before Plastic SCM can be used.

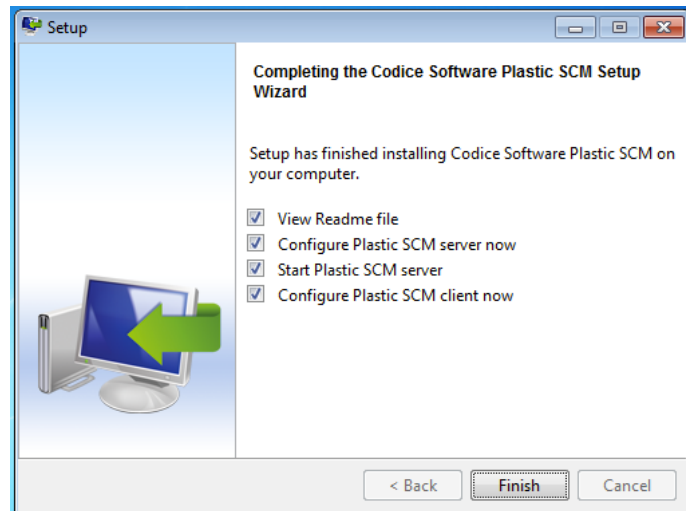


Figure 5. Start system configuration

Note: If the Windows Explorer Shell Extension has been installed, a reboot of the machine might be needed.

Table 1. Installation steps

3.3 Server configuration

The server configuration wizard can be started at the last step of the installer, or from the Plastic SCM startup menu entry.

Server configuration start up screen.

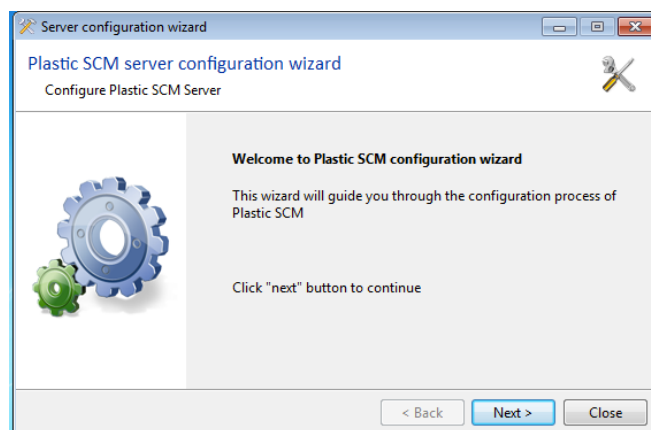


Figure 6. Server configuration welcome screen

Select the server's language. Used for logging and errors coming from the server to the clients.

Clients and server can have different languages. Clients will get messages localized in their own configured language.

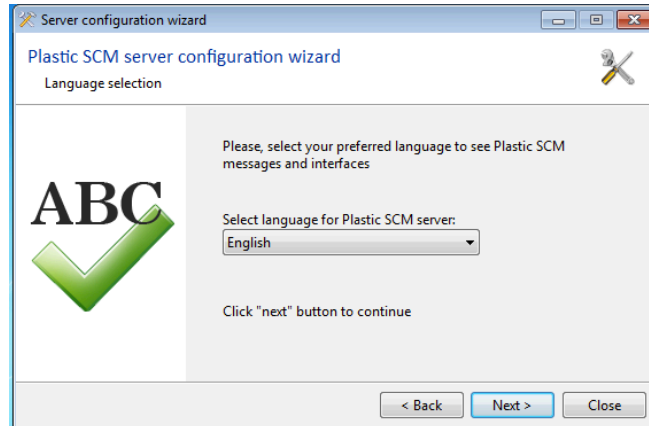


Figure 7. Server configuration language selection

Authentication mechanism. This is the most important step in the server configuration wizard.

Here you have to choose between the different authentication mechanisms available. By default *local users* is selected.

More details on the authentication modes can be found on section 3.4 below.

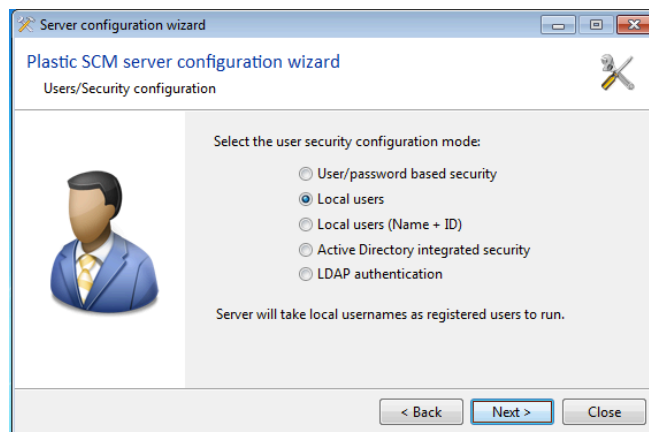


Figure 8. Server configuration authentication modes

If you choose the LDAP user security mode, you must fill in some more parameters.

You have to specify the LDAP server name, and the domain from which data must be retrieved.

Also a username and a password to be used to access the LDAP server.

Finally choose whether the LDAP server is a conventional LDAP one or an Active Directory.

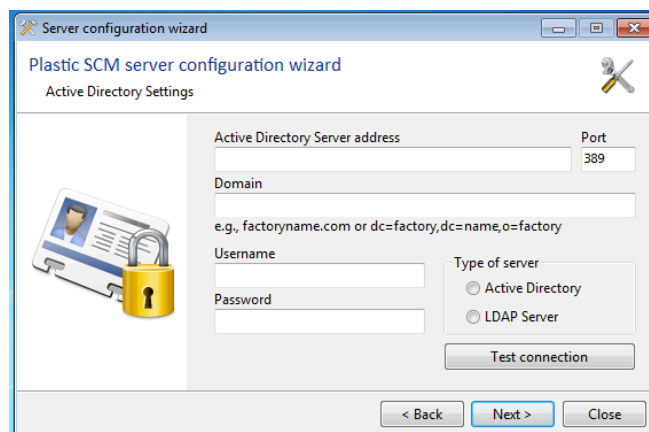


Figure 9. LDAP Authentication configuration

The last step is configuring the server's TCP port number. By default Plastic SCM will use 8087 as its TCP port number.

If the default listening port is changed, Plastic SCM clients must be setup to this new port.

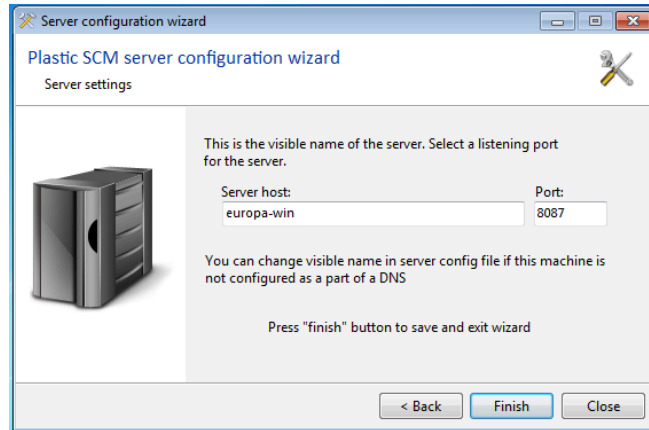


Figure 10. Port configuration

Table 2. Server configuration

3.4 User authentication configuration

Authentication methods tell Plastic SCM how to integrate users and groups of users with the objects of the repository. Plastic SCM can use five different connectors for retrieving its user information:

- Local users of the machine (Only name)
- Local users of the machine (Name + ID)
- Integrated with Windows Active Directory
- LDAP
- Plastic SCM's own User – Password database

Each of them allows different authentication possibilities and will be explained in the following sections.

3.4.1 Authentication basics

Client communicates certain security information to the server in order to be validated. The basic token sent from client to server is called SEID, the short name for SEcurity IDentifier.

The mechanisms to be described are basically based on different ways to build the SEID plus different ways to obtain users.

3.4.2 Local users

In Local users mode, the Plastic SCM server will read the local users names from the machine it is running on. So on startup it will create a list of *known users*, and recalculate it periodically.

For the system to work correctly the Plastic SCM clients must also be configured using the *Local Users* mechanism.

The client will take the name of its logged on user and send it to the server. This is the name that the server will use to first check whether it is a known user, and then make security calculations with.

This system relies on correct network configuration. It can be used on secured networks to easily configure a mixed Unix / Windows environment, relying, for instance on a NIS+ system.

It can also be used for easily configuring access from the Internet, provided that the server only allows trusted clients to connect.

How does the server obtain the user list?	It retrieves it from the local machine users (both Unix and Windows operating systems). For Windows machines inside a domain it will take the current user if it's not a local user.
How is the SEID built?	Just with the user name.

3.4.3 Local users: name + ID

The mechanism is identical to *local users* but the SEID is built using both the user name plus the user ID.

It is a very simple way to prevent, or at least complicate a bit further, identity hijacking.

Under Windows systems the ID will be the SID of the user.

Under Unix based systems it will be the user id.

It works perfectly on non-cross-platform environments (Unix-Unix or Windows-Windows) but it will obviously break under Windows-Unix platforms unless a specific authentication mechanism is in place.

It can be used to work under NIS+ systems on Unix, or under any other configuration provided that both systems share the same user name and ID.

How does the server obtain the user list?	It retrieves it from the local machine users (both Unix and Windows operating systems). For Windows machines inside a domain it will take the current user if it's not a local user.
How is the SEID built?	User name + ID: user id on Linux and SID on Windows.

3.4.4 Active Directory Integrated Security

Using this configuration mechanism the user list will be retrieved from the current Active Directory server. This system needs the server to be running inside an operating system that can be part of an Active Directory. It is designed to run on Windows based operating systems.

How does the server obtain the user list?	It retrieves it from the active directory main server. The server must be correctly configured.
How is the SEID built?	A Windows SID.

3.4.5 LDAP

The LDAP security configuration mechanism allows interoperability with an LDAP environment.

It can be used to authenticate users against any kind of LDAP server. A Sun One or iPlanet LDAP Server can be used, for instance, to authenticate Plastic SCM users.

This is also a good method for Windows / Unix mixed environments, since Plastic SCM can connect to an Active Directory server using the LDAP mechanism, for instance when connecting from a Unix box where the integrated Active Directory mode is not available.

How does the server obtain the user list?	From the LDAP Server using a given user and password.
How is the SEID built?	The ID used by the concrete LDAP mechanism.

3.4.6 User/Password

The user/password authentication system introduces an easier way to configure Plastic in a user - group based authentication in certain environments. When LDAP, Active Directory and local server authentication mode are not available options, the system administrator can select user/password authentication.

When using UP mode (meaning user/password authentication mode) the Plastic SCM security system works exactly as it would do with LDAP, Active Directory or any other mode, the difference being that the Plastic SCM server itself will store the user and group information.

When working in UP mode the administrator will define users, groups and their relationships using a specific Plastic configuration tool, then the client will just have to specify the previously configured user and password to log in the Plastic server.

UP authentication mode is appropriate for mixed Linux/Windows environments where LDAP or Active Directory integration is not an option, or to manage access to the Plastic SCM server on heterogeneous environments where there is no common user login among operating systems.

To configure the login and password the user needs to run the Plastic SCM configuration wizard as shown on the following Figure 11.

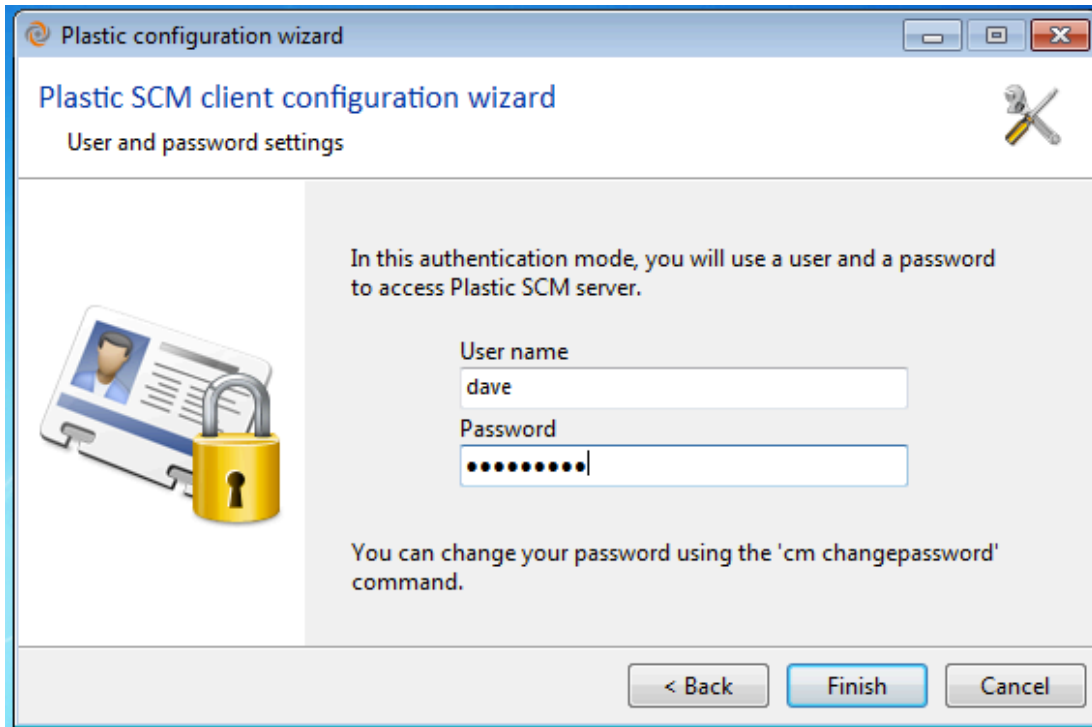


Figure 11. User and password configuration screen

When the Plastic SCM GUI client starts up, a login screen will pop up if the user or password doesn't match.

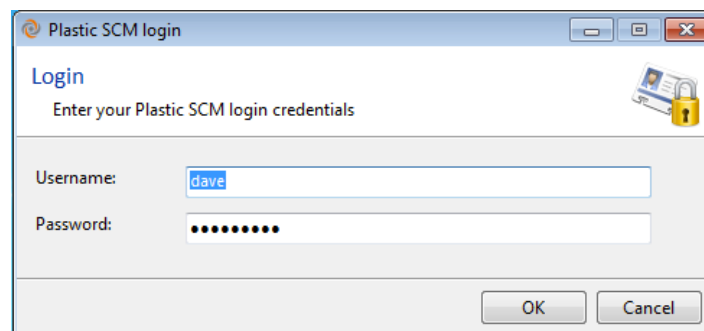


Figure 12. Login dialog

The main difference between UP and the other authentication methods is that instead of relying on an external user and group *provider*, the UP authentication mode stores all its data into two files: *users.conf* and *groups.conf*.

- **users.conf**: stores information about all the users and their encrypted passwords.
- **groups.conf**: stores all the available groups and the users they contain.

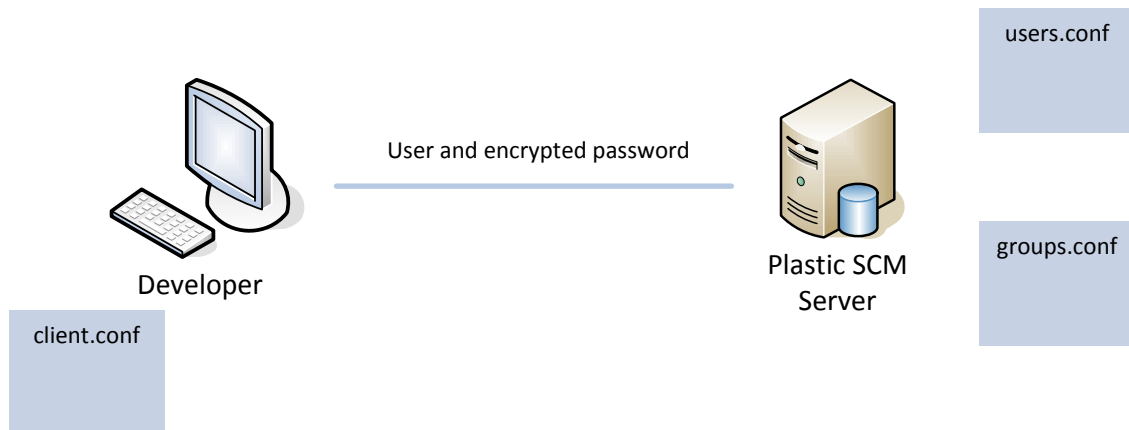


Figure 13. configuration files in user/password mode

Configuring user/password authentication mode

In order to configure the UP mode you'll use the following tools:

- The server's configuration tools (**configureserver** or **clconfigureserver**) to set the authentication mode of the server.
- The client's configuration tools (**plastic --configure** or the configuration wizard) to specify the authentication mode used to communicate with the server.
- **umtoolgui** or **umtool** to configure the users, groups, and their relationships both graphically and from the command line.

Selecting the internal user/password authentication mode

To select Plastic SCM's user/password authentication mode it has to be the selected authentication mode in the server and the client. This is done through the configuration wizards.

In the client, the configuration wizard can be manually launched using the "client configuration wizard" shortcut in the startup menu, in *Plastic SCM / client tools*, or using the command line:

```
plastic --configure
```

On the server, the configuration wizard can also be found as a shortcut in the startup menu, under *Plastic SCM / server tools / Server configuration wizard*, or using the command line:

```
configureserver
```

3.4.6.1 Managing users in user/password mode

Umtoolgui is the GUI tool to configure the users, groups and their relationships and passwords. The tool is located on the server's installation directory.

Figure 14 illustrates all the **umtoolgui** options. It is a simple and intuitive tool whose solely purpose is to help users configure the `users.conf` and `groups.conf` files.

The tool is able to create users and groups, assign users to a specific group, change a user's password, and rename or delete users and groups.

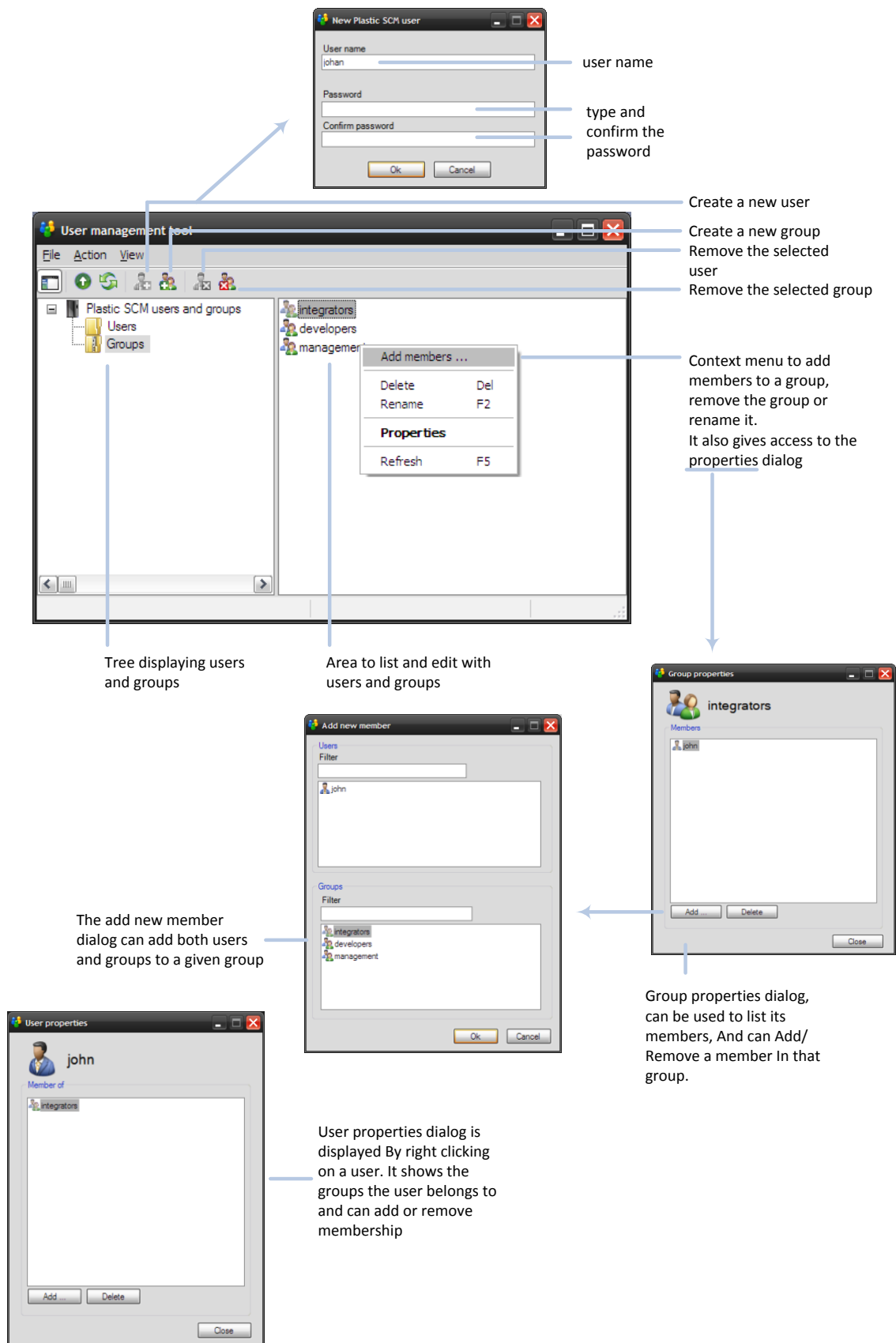


Figure 14: umtoolgui usage

Umtool is the command line tool used to configure the users, groups and their relationships and passwords from the operating system's console.

umttool implements several commands detailed on the following Table.

Command name	Short name	Description	Syntax
addgrouptogroup	agtg	Include a new group into a group	umttool addgrouptogroup <grouptoadd> <groupname>
addusertogroup	autg	Include a new user into a group	umttool addusertogroup <username> <groupname>
changeuserpassword	cup	Change a user's password	umttool changeuserpassword <username> <oldpassword> <newpassword>
creategroup	cg	Create a new Plastic SCM group	umttool creategroup <groupname>
createuser	cu	Create a new Plastic SCM user	umttool createuser <username> <password>
deletegroup	dg	Delete a existing Plastic SCM group	umttool deletegroup <groupname>
deletegroupfromgroup	dgfg	Delete a group from a group	umttool deletegroupfromgroup <grouptodelete> <groupname>
deleteuser	du	Delete a existing Plastic SCM user	umttool deleteuser <username>
deleteuserfromgroup	dufg	Delete a user from a group	umttool deleteuserfromgroups <username> <groupname>
help	hlp	Show a command's help	umttool help <commandname>
listgroupmembers	lgm	Show a list with members of a group	umttool lgm <groupname>
listgroups	lg	Show a list with current Plastic SCM groups	umttool lg
listusers	lu	Show a list with current Plastic SCM users	umttool listusers
renamegroup	rg	Rename a existing Plastic SCM group	umttool renamegroup <oldgroupname> <newgroupname>
renameuser	ru	Rename a existing Plastic SCM user	umttool renameuser <oldusername> <newusername>

Table 3 umttool commands

users.conf file format

The users.conf file contains the definition of the users known to the system in user/password authentication mode. The format of the users.conf file is very simple: it contains a list of the available users followed by their passwords as shown on the figure below.

users.conf configuration file

```
john: 519c84155964659375821f7ca576f095
paul: 32252792b9dccb239f5a5bd8e778dbc2
daniel: 63d658d767c4292849aa031434e3d060
charlie: 7e4b64eb65e34dfad79e623c44abd94
mike: 3492aa3efe4d423f873607e7e45a0d7e
peter: 9412aa3efadfa73ad07e7e45a0d7a
johan: 13d658d767c4292849aa031434e3d060
```

user name
encrypted password

Figure 15. users.conf file format

groups.conf file format

The groups.conf file contains all the groups known to the Plastic system in user/password mode. The file is a list of the groups, each one followed by the names of the users or groups it contains. Check the following figure for details.

groups.conf configuration file

```
integrators: john:paul:daniel:
developers: john:charlie:mike:peter:
management: johan:
```

group name
group members

Figure 16. groups.conf file format

Setting server's UP working mode modifying server.conf

After the server's working mode is set, it will be stored on the configuration file named server.conf.

The server.conf file can be manually modified to choose a different authentication mode. To set the user/password working mode, use the *workingmode* value specified in the figure below.

```
<ServerConfigData>
  <Language>en</Language>
  <WorkingMode>UPWorkingMode</WorkingMode>
  <SecurityConfig></SecurityConfig>
</ServerConfigData>
```

UPWorkingMode can be set manually by editing server.conf file on the server's directory

Figure 17. server.conf contains WorkingMode

User/Password mode common questions

How does the Server build the list of users?	It retrieves the list of users' names from the users.conf and the groups.conf files on the server folder.
What does the authentication contain?	It contains the username and the encoded password.

This is the traditional authentication method; it allows Plastic SCM users to define their own users and groups on the Plastic server. This way Plastic can work with an autonomous security mechanism, which could be the best option for many organizations that don't rely on systems like LDAP or Active Directory.

The server keeps a list of the users and each user defines his password. It also keeps groups as well as the relation between users and groups. Each client sets a user and a password in order to have access to the server, the user must exist on the server and the password must be the same one.

The list of users and groups is defined by two configuration files located in the server folder.

There are two tools used to manage users and groups, these tools can be found on the server's installation directory:

umtool (command line tool)

umtoolgui (Graphic User Interface tool)

3.5 Server startup

Plastic SCM server starts automatically on server boot and after the installation process is finished, but it can be stopped, started, or restarted manually.

3.5.1 Windows Systems

To start and stop the Plastic SCM service on Windows Systems, you can open the Windows Service Manager. Go to Control Panel → Administrative Tools → Services. There you will find the Plastic SCM service in the list. See Figure 18.

Alternatively you can go to START → RUN and type: services.msc to open the windows services window.

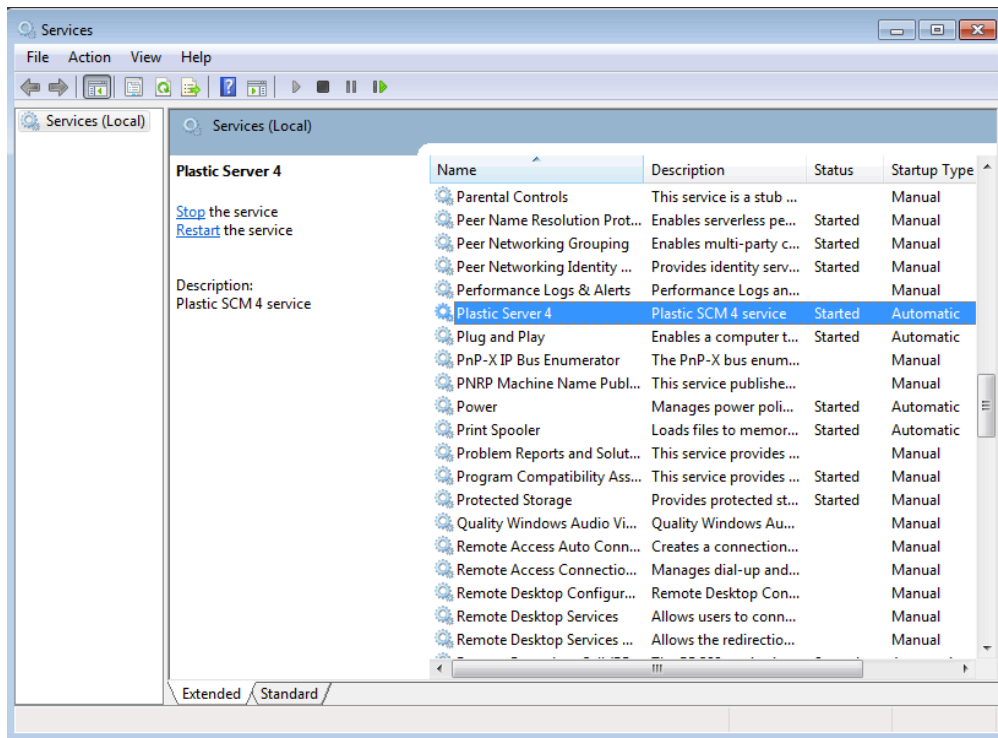


Figure 18. Managing Plastic SCM server service

You can use the start and stop actions available to normal services from the Services console.

3.5.2 Linux Systems

To start and stop the Plastic SCM service on Linux Systems, you can use the *plasticsd* script. This script is located in the server installation directory (*/opt/PlasticSCM/server* by default). A script *plasticd* is also installed on the */etc/init.d* directory to help with automatic start on system boot. On RedHat based systems you can use the program *service plasticsd <options>*. This script has the following options:

```
./plasticsd {start | stop | restart | status}
```

Once the Plastic SCM server is started, a *default* repository will be created, to ease the initial system usage.

To check whether the server is up and running, simplest way:

```
/etc/init.d/plasticsd status

PlasticSCM server is started (PID xxxx)
```

PID xxxx is the running process ID of the server.

Another way to check the server status is by looking at the repository list. Follow the next steps:

Open a console and type:

```
cm lrep servername:port
```

It will list all the repositories on server *servername:port*.

3.6 Client configuration

Each time a new client is installed on a developer machine, it must be configured to enable it to connect to a Plastic SCM server. This can be easily done using the client configuration wizard. The steps in the wizard are explained below.

Client configuration welcome screen.

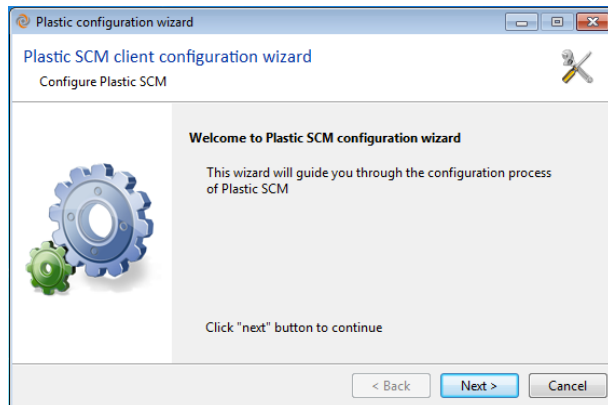


Figure 19. Client configuration welcome screen

Language selection.



Figure 20. Client configuration language selection

Fill in the host name or IP address of the Plastic SCM server

Optionally you can use a proxy server.

For more information regarding Plastic SCM proxy server see chapter 4 Using a proxy server

By default, the Plastic SCM server TCP port is 8087. If it has been changed on the server, set here the new port number.

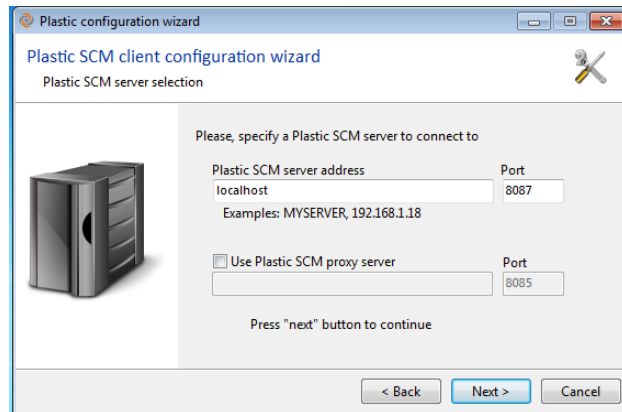


Figure 21. Workspace server selection

Next step is to choose an authentication mechanism as used in the Plastic SCM server.

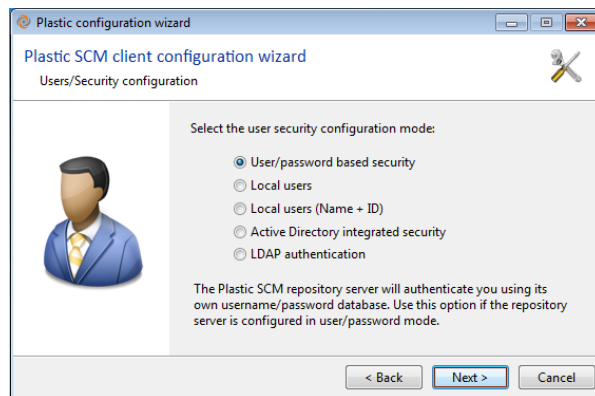


Figure 22. Client authentication selection

If Active Directory integrated security was Plastic server configuration, the client can choose between that same mode or LDAP authentication.

For Unix clients, this is the way of connecting to an Active Directory based server.

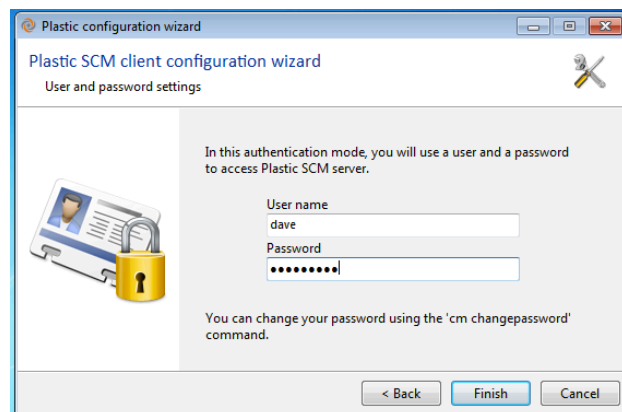


Figure 23. Client AD LDAP authentication window

Table 4. Client configuration steps

The client is now ready to be used and can be started from the startup menu or by typing *plastic* in a command line interface window.

3.6.1 Merge and Differences Tools configuration

This section describes how to configure the Plastic SCM client to use a specific merge or diff tool for specific types of files. Note that Plastic SCM includes its own 3-way merge and diff tool, so this step is not required in the default setup.

The configuration of the merge and differences tools are defined in the *client.conf* file. It allows specifying what tools have to be used for different types of files through a set of rules.

A rule contains information regarding the type of file (binary or text) and optionally the file extension to which this rule applies.

The default rules for the merge tool are listed here:

- **FileType:** Indicates the types of files the rule will apply to, they can be *TextFile*, for files identified as text by Plastic SCM or *BinaryFile* for files identified as binary by Plastic SCM.
- **FileExtensions:** Indicates file extensions or types of files on which the rule is applied, if we have more than one extension for the same rule they would be separated by ";". If the rule is used for every extension "*" would be used.
- **Tools:** Different merge tools to be used on a specific file, if there is more than one tool, they are executed in order until one of them gives a result; only the first tool is used in the case of the differences. These tools must have every mandatory parameter using the variables given, which are replaced by the system value during execution. If we want the system to use a specific value for a rule instead of a variable, that value would be set. The given variables are the following ones:
 - **@basefile:** path containing the merge common ancestor.
 - **@basesymbolic:** name shown on the tool to refer to the base file, it is usually the spec revision or the disc file if loaded.
 - **@basehash:** common ancestor content hash.
 - **@sourcefile:** path containing the merge source file.
 - **@sourcesymbolic:** name shown on the tool to refer to the merge source file, it is usually the spec revision or the disc file if loaded.
 - **@sourcehash:** merge source content hash.
 - **@destinationfile:** path containing the merge destination file, the location of the element in the workspace.
 - **@destinationhash:** merge destination content hash.
 - **@output:** file containing the merge result.
 - **@filetype:** type of file used for the syntax highlight
 - **@comparisonmethod:** comparison method used.
 - **@fileencoding:** file encoding.
 - **@mergetype:** type of merge used.

It must be highlighted that rules are executed in order, so the less restrictive ones must be at the bottom of the list. A catch-all rule for the file extension "*" is normally the latest.

An example below shows using an additional rule for a single type of file considered as binary (.scs) if we want it to be considered as text and given fixed parameters: type of merge to be automatic only if one of the contributors has submitted changes and file codification as Unicode:

```

<MergeTools>
  <MergeToolData>
    <FileType>enTextFile</FileType>
    <FileExtensions>*</FileExtensions>
    <Tools>
      <string>mergetool -b="@basefile" -bn="@basesymbolic" -
bh="@basehash" -s="@sourcefile" -sn="@sourcesymbolic" -
sh="@sourcehash" -d="@destinationfile" -dh="@destinationhash" -a -
r="@output" -t="@filetype" -i="@comparationmethod" -
e="@fileencoding" -m="@mergetype"</string>
    </Tools>
  </MergeToolData>

  <MergeToolData>
    <FileType>enBinaryFile</FileType>
    <FileExtensions>.scs</FileExtensions>
    <Tools>
      <string>"mergetool.exe" -b="@basefile" -bn="@basesymbolic"
-bh="@basehash" -s="@sourcefile" -sn="@sourcesymbolic" -
sh="@sourcehash" -d="@destinationfile" -dh="@destinationhash" -a -
r="@output" -t="@filetype" -i="@comparationmethod" -e="unicode" -
m="onlyone"</string>
    </Tools>
  </MergeToolData>
  <MergeToolData>
    <FileType>enBinaryFile</FileType>
    <FileExtensions>*</FileExtensions>
    <Tools>
      <string>binmergetool -b="@basefile" -bn="@basesymbolic" -
bh="@basehash" -s="@sourcefile" -sn="@sourcesymbolic" -
sh="@sourcehash" -d="@destinationfile" -dh="@destinationhash" -a -
r="@output" -m="@mergetype"</string>
    </Tools>
  </MergeToolData>
</MergeTools>

```

In the case of the differences tools, rules are built similarly but there are fewer variables. In order to add a rule to calculate differences on a binary file (.scs) as a text file with Unicode format, we would write the following:

```

<DiffTools>
  <DiffToolData>
    <FileType>enTextFile</FileType>
    <FileExtensions>*</FileExtensions>
    <Tools>
      <string>mergetool -s="@sourcefile" -sn="@sourcesymbolic" -
d="@destinationfile" -dn="@destinationsymbolic" -a -t="@filetype"
-i="@comparationmethod" -e="@fileencoding"</string>

```

```
</Tools>
</DiffToolData>
```

```
<DiffToolData>
  <FileType>enBinaryFile</FileType>
  <FileExtensions>.scs</FileExtensions>
  <Tools>
    <string>"mergetool.exe" -s="@sourcefile" -
sn="@sourcesymbolic" -d="@destinationfile" -
dn="@destinationsymbolic" -a -t="@filetype" -
i="@comparationmethod" -e="unicode" </string>
  </Tools>
</DiffToolData>
<DiffToolData>
  <FileType>enBinaryFile</FileType>
  <FileExtensions>*</FileExtensions>
  <Tools>
    <string>binmergetool -s="@sourcefile" -
sn="@sourcesymbolic" -d="@destinationfile" -
dn="@destinationsymbolic" -a -t="@filetype" -
i="@comparationmethod" -e="@fileencoding"</string>
  </Tools>
</DiffToolData>
</DiffTools>
```

4 Using a proxy server

Plastic SCM supports the idea of having a Proxy server to support balancing the traffic between the two machines: the possibly remote server machine and the probably local proxy machine. This proxy can be installed with minimum operating resources. No database backend is needed to run it, making it pretty easy to set up. It can be configured as a daemon, Windows service or started manually from the command line.

The cached data will always be read-only, meaning that only checked-in revision data is cached on the proxies. Since checked in revisions never change, it's safe and simple to cache them.

The Proxy server will start caching files from the repository server as data is requested from clients. Once cached, revision data stays on the proxy, so that future requests to the same revisions will be downloaded from the proxy machine instead of the remote repository server.

The proxy server will also use keep the most used revisions in memory, for improved performance. The administrator can configure the maximum amount of memory to be used.

This is a list of the benefits of using a Proxy with Plastic SCM:

- The server will act as a data cache. No other data or metadata besides revision information will be stored.
- All cached data will be stored on a configurable disk directory for simplified administration.
- Data transmission between the proxy server, repository server and client will use the same data exchange protocol used by Plastic SCM, exchanging data chunks of 4Mb max size.

The following is an example of a configuration using a proxy server:

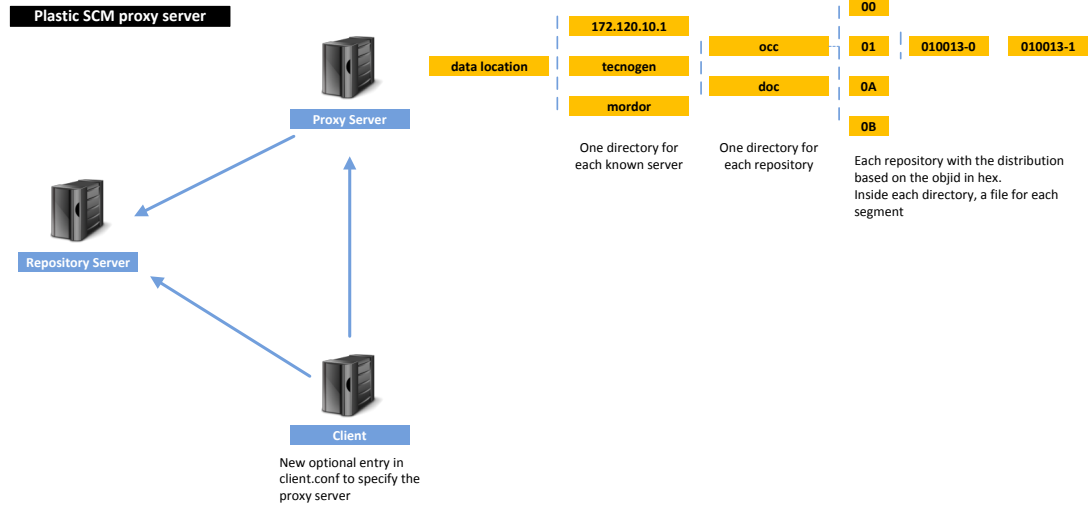


Figure 24. Example of proxy deployment

The basic usage scenario is depicted on the following set of figures. The typical scenario for *proxy server* is a centralized setup over a slow network, which needs to be optimized.

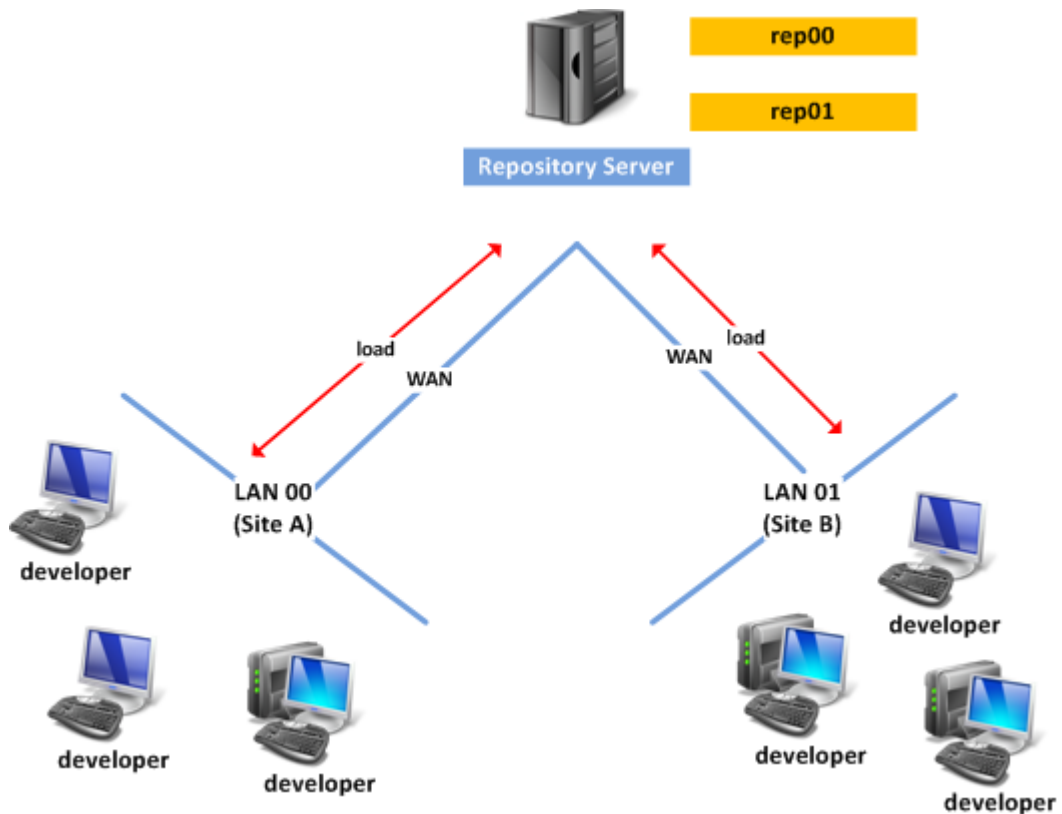


Figure 25. Example of network

Each site can take advantage of a very simple *proxy server*, which will increase the overall system performance by reducing network traffic over the distant networks.

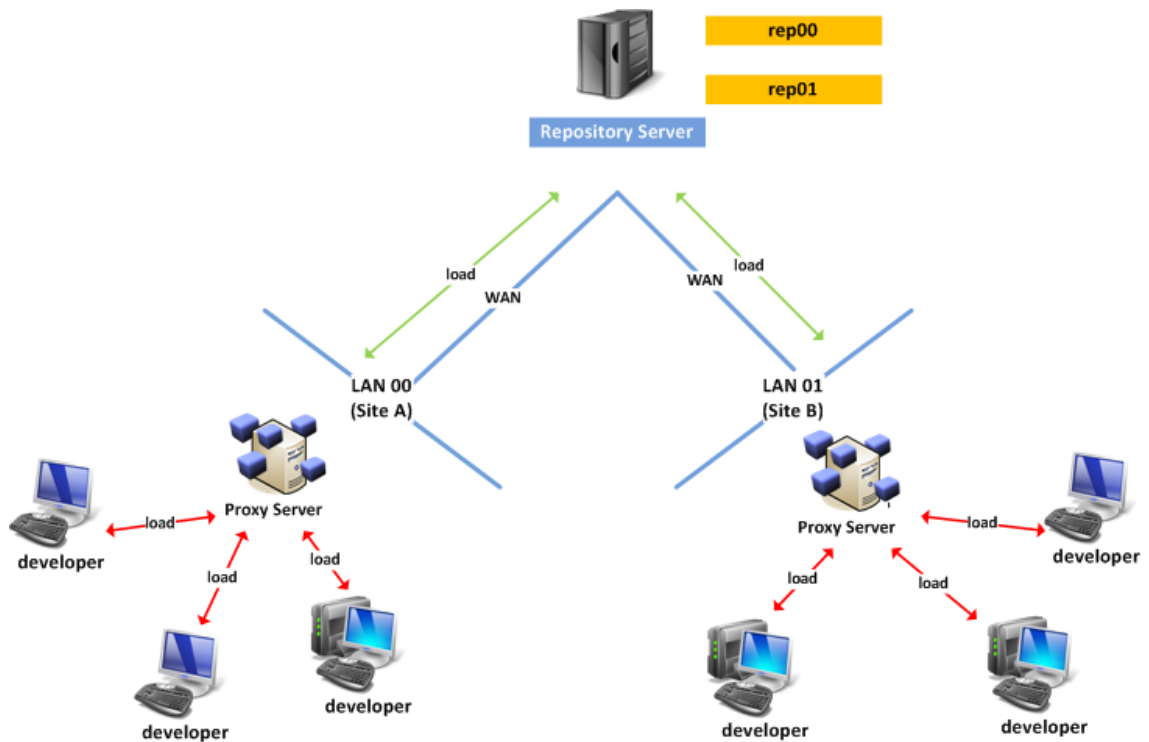


Figure 26. Same network, improved by using two proxies

4.1 Installing the Plastic SCM Proxy server

The Proxy server is distributed as a separate download in the Plastic SCM website (www.plastic SCM.com) and can be installed stand alone: it doesn't require a Plastic SCM server or client to be present on the same machine.

The following table summarizes the installation steps after the installer is run and the license agreement accepted:

Installation directory selection.

This is the folder that contains the binary files for the proxy server. The next step lets you configure the directory where the cached data will reside.

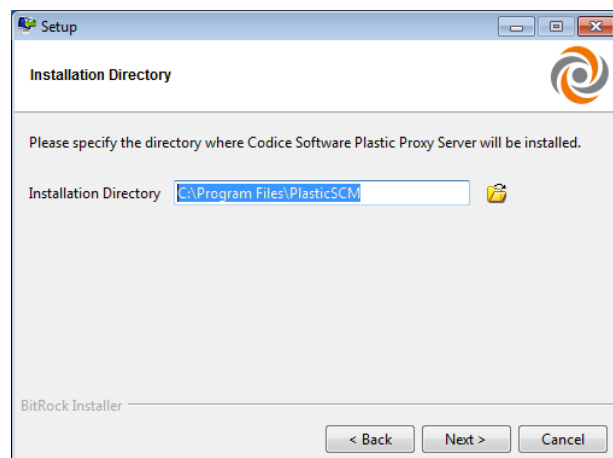


Figure 27. Proxy installer binaries folder

Cache storage.

This is the folder that will hold the revision cache. It should be located on a fast and large enough disk, to benefit from the maximum performance.

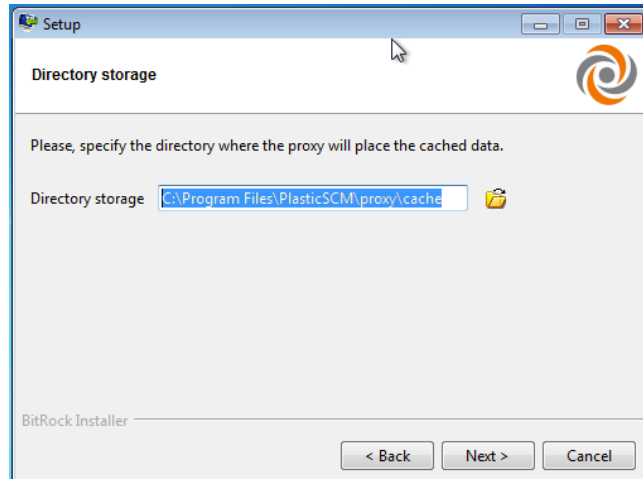


Figure 28. Cached data directory location

TCP Listening port.

This is the TCP port that the proxy server will use. By default, it is 8085.

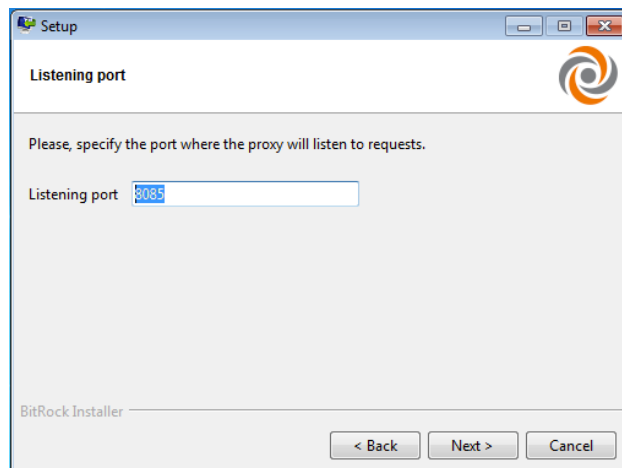


Figure 29: Proxy listening port

Binary files will be copied and the installation is complete.

The Plastic Proxy service is started at the end of the installation and is ready to begin caching data.

4.2 Configuring the clients

The clients need to configure a proxy server in order to benefit from its cache. This is done in the Plastic SCM client configuration wizard, in the server page. The proxy server is configured by checking the "Use Plastic SCM proxy server" box and entering the proxy server host address and port in the textboxes.

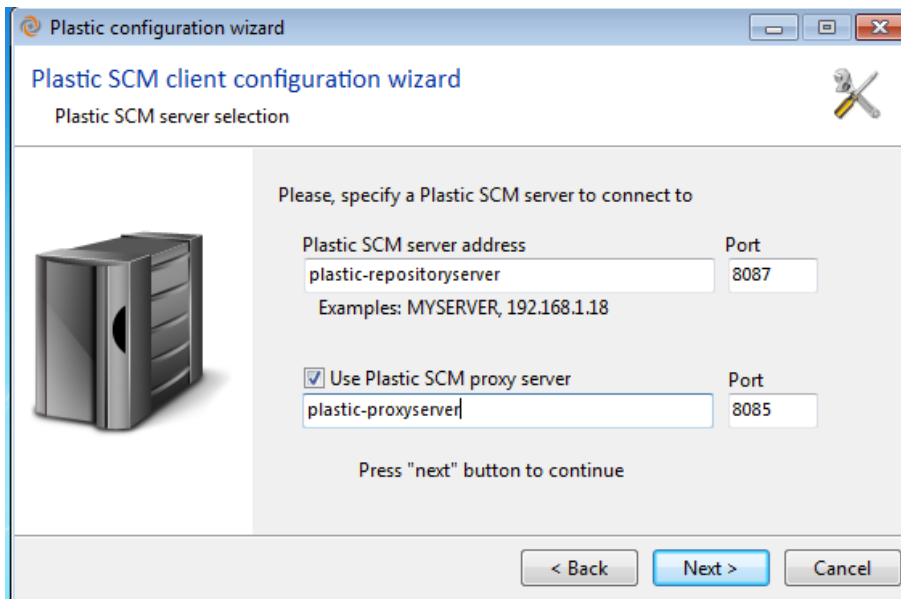


Figure 30: configuring a proxy server in the Plastic SCM client

This is all the setup required in the client. When an operation downloads revisions from the server, the client will ask the proxy and download from it instead of the server if the data is cached.

5 Creating and managing repositories

Repositories are the central data storage in the Plastic SCM system. They store the information for all the objects in the system.

5.1 Creating a new repository

An empty repository named *default* is created on the first server start up if no other repository exists yet, so that users can start working straight away.

From the administrator point view, it can be desirable to create repositories using the command line interface. This is achieved in with the `cm more` command:

```
cm mkrep PlasticServer:8087 NewRepository
```

Where:

- PlasticServer is the hostname or IP address of the Plastic SCM server.
- 8087 is the TCP port where the server is listening.
- NewRepository is the name of the new repository to be created.

Repositories can also be created on the Plastic SCM GUI client by opening the repositories view and clicking on "Create new repository".

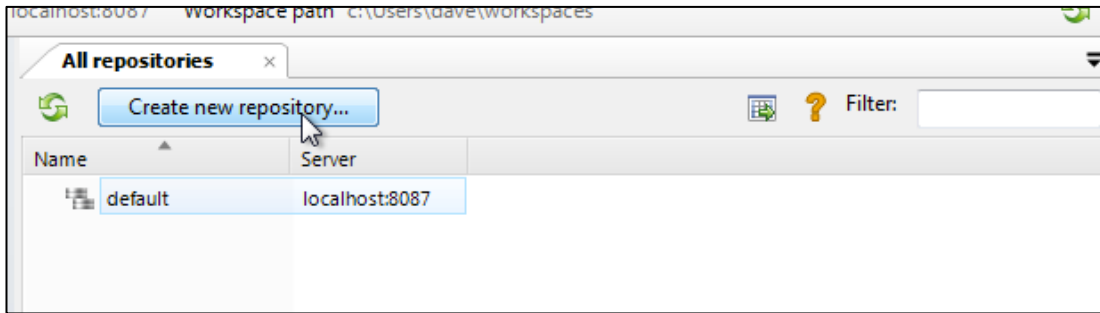


Figure 31: creating a new repository in the Plastic SCM GUI client

The new repository dialog lets the user enter the repository name and the Plastic SCM where it will be created.

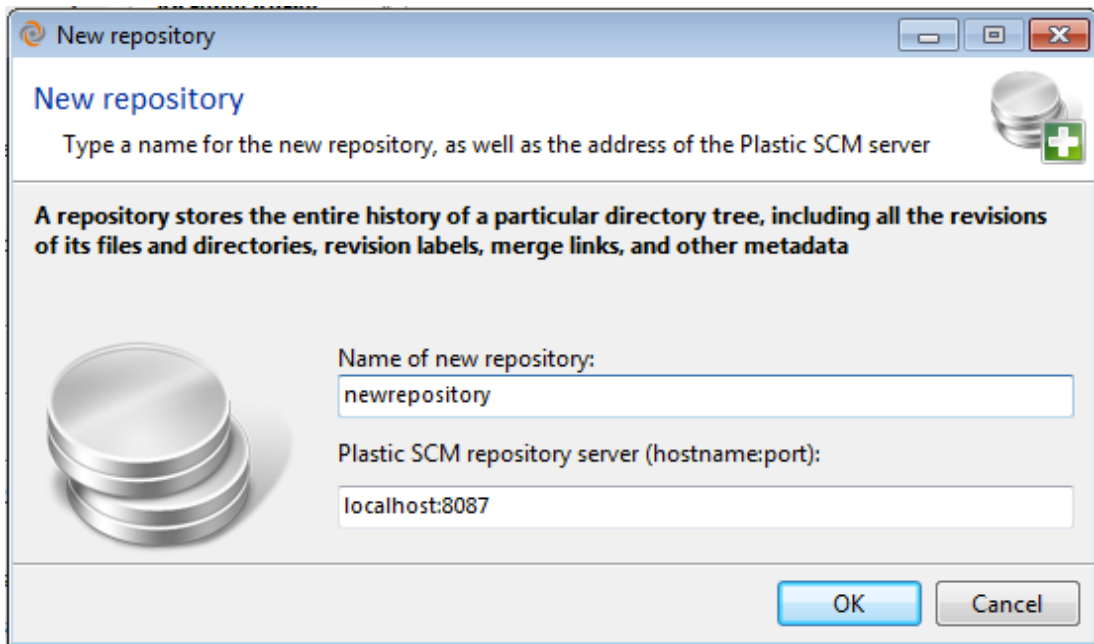


Figure 32: New repository dialog

5.2 Listing available repositories

Listing available repositories can be done at the command line level or GUI client. On the command line, you use the `cm listrepositories` command, or `lrep`:

```
cm lrep PlasticServer:8087
```

Will show the this output:

```
1 default PlasticServer:8087
```

Figure 33 shows the repositories view on the GUI tool, accessible from the *View* menu.

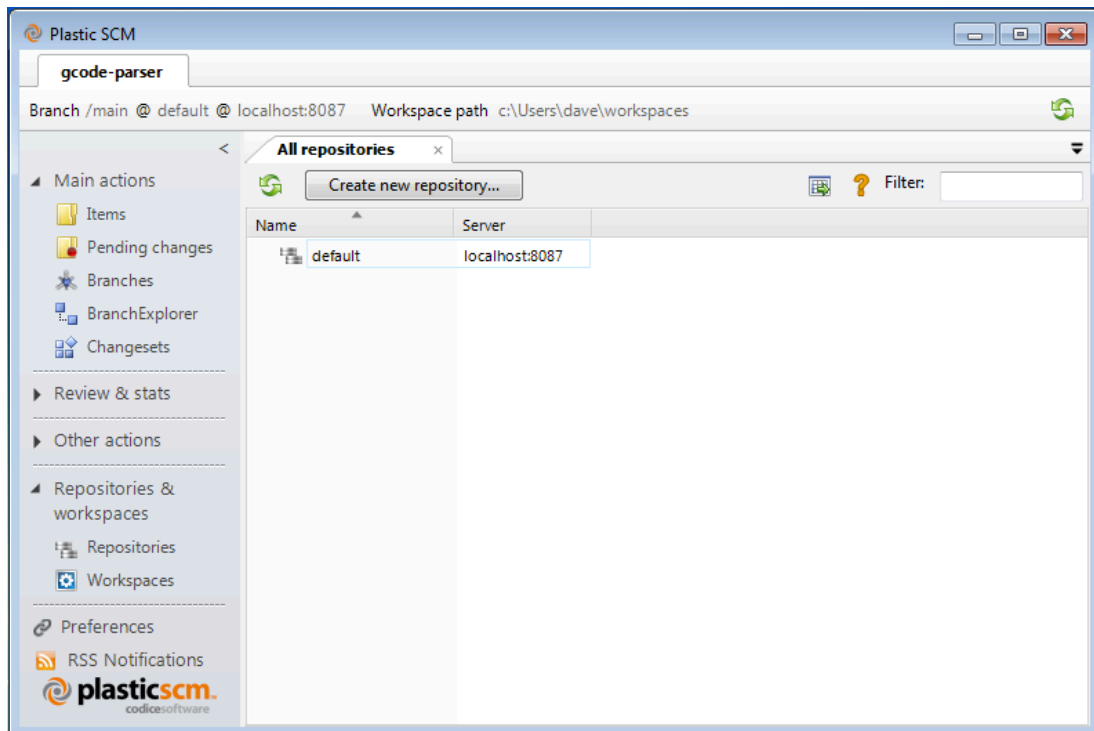


Figure 33. Repositories view

5.3 Archiving repositories

Repositories that are no longer used can be disconnected from the server and archived on offline storage like DVD or tapes, leaving free space for active repositories. These repositories can be later reconnected if they need to be used, following the procedures described here.

In order to archive a repository, the *removerepository* command is used, providing a repository specification. First, list all repositories:

```
C:\scm3>cm lrep localhost:8087
1 default localhost:8087
3 myproject localhost:8087
5 nasa localhost:8087
```

Get the number in the first column for your repository "the repository index". In the sample above, we want to archive repository 3 (myproject). This number will be used later to identify the file to backup.

Next is removing the repository:

```
cm removerepository myproject@localhost:8087
```

This command has two internal effects, important to note for later reconnection.

- a) It removes the repository from the list of registered repositories available to be used in workspaces.
- b) It removes the repository from the list of available repositories.

Note: the `removerepository` command does **not** delete the database in the database backend.

The repository database in the physical database backend name follows this pattern:

```
rep_xx.plastic
```

The file for *myproject* repository, in the previous sample, will be

```
rep_3.plastic
```

You can locate the database with that name in your database backend and back it up using the tool of your choice. After you have backed it up, you can delete the database from your backend if desired.

5.4 Reconnecting archived repositories

To reconnect an archived repository, first get a list of current repositories (this might be different from the list of reps found when it was archived)

```
C:\scm3>cm lrep localhost:8087

1 default localhost:8087
3 excel localhost:8087
4 word localhost:8087
5 PW point localhost:8087
6 access localhost:8087
7 outlook localhost:8087
```

In the sample above, several repositories have appeared/created since we archived *myproject*, which had number 3. Choose a free repository number not used in the list (next one is 8, but you can choose any other). Rename your database `rep_3.plastic` to `rep_8.plastic`. For instance, for an embedded Firebird database this means just renaming the database file on disk:

```
move rep_3.plastic.fdb rep_8.plastic.fdb
```

And now for reconnecting, add the repository to the list of available repositories using the `cm addrepository` command:

```
cm addrepository rep_8 myproject localhost:8084
```

The first argument is the database name, without the `.plastic.fdb` extension.

The second argument is the name of the repository. This must be unique in the list of available repositories and is the name that Plastic SCM users will see.

The last argument is the repository server IP:TCP PORT where the repository will be connected.

6 Database setup

Plastic SCM 4 support several database backends to store repository data. This is the list of supported backends:

- MySQL
- SQL Server & SQL Server Express
- SQL Server Compact Edition
- Oracle
- Firebird Server
- Firebird Embedded
- SQLite
- PostgreSQL

After the first installation, the Plastic SCM server will use an embedded database backend: SQL Server Compact Edition if the server is running on Windows, or SQLite is running on Linux.

The database backend can be changed using the Administration Tool, accessible from the startup menu in windows, under the Plastic SCM 4 menu / Server.

To change the database backend, click on *Database* on the left panel and then click the "*Migrate database backend*" button on the right panel. This operation will change migrate your existing repository data from the current database backend to the new one of your choice.

Note: please note that migrating your data requires stopping the Plastic SCM server and it can take some time if your repositories have lots of data. While the migration is running, your users will not be able to access the Plastic SCM server.

If you have just installed Plastic SCM and have no data in your repositories yet, the migration is very fast.

Figure 34 shows the migration wizard initial screen, with a Plastic SCM server currently configured to use the default SQL Server CE backend (on the top) and prompting the user to choose the new database backend.

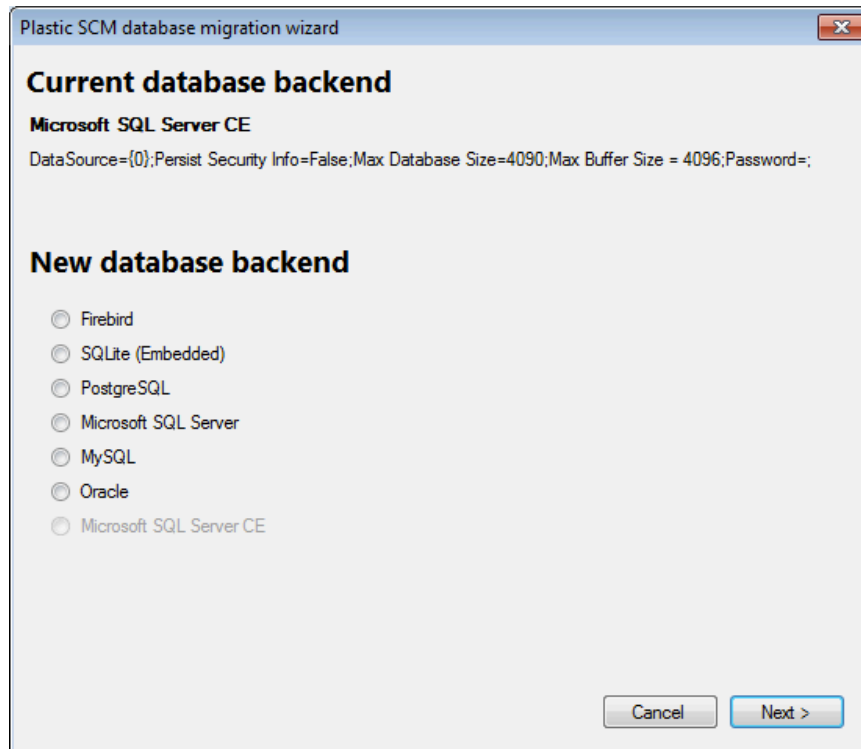


Figure 34: Database migration wizard

Selecting a database backend and clicking next will move the wizard to the second step, where the options for the database are filled.

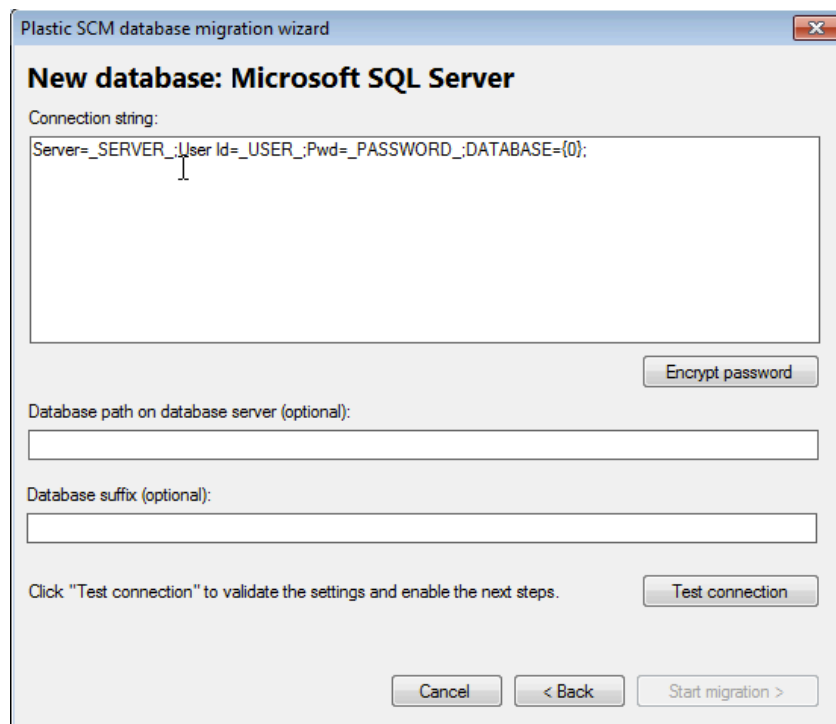


Figure 35: database migration: target database options

This is a list of the options common for all backends:

Connection string: the most important option and the only mandatory one: specifies the server, credentials and other flags associated to the connection. The wizard provides a default connection string for all the backends it supports, with the needed flags. In general, you only need to fill in the database server name, the user and the password to connect to the database backend.

Note that Plastic SCM will need to create several databases on the database backend, so the credentials supplied need to be granted database creation permissions.

The following table covers the supported backends with their default connection strings and the arguments that you need to replace in the connection string with your own values:

Backend	Arguments
Firebird	By default, Firebird uses the embedded backend, which doesn't require any specific arguments. The default user and password are used to connect to the embedded server.
SQLite	By default, SQLite uses the embedded backend, which doesn't require any specific arguments.
PostgreSQL	_SERVER_: the database server machine. _USER_: user account to connect to the backend _PASSWORD_: password for the account.
MS SQL Server	_SERVER_: that database server machine and instance. _USER_: user account to connect to the backend, if using SQL Server authentication. _PASSWORD_: password for the account
MySQL	_SERVER_: the database server machine. _USER_: user account to connect to the backend _PASSWORD_: password for the account.
Oracle	_SERVER_: the database server machine _ORACLE_SID_: the oracle_sid of the database to connect to. Oracle has more specific options in this dialog that are described below.
MS SQL Server CE	By default, SQL Server CE uses the embedded backend, which doesn't require any specific arguments.

Database path on the server: optionally, you can set the directory where the Plastic SCM will ask the database backend to store the database physical files.

It is common to have a database server with a faster or bigger secondary disk (different than the one used for the system). Normally you can specify where to create databases when you create them in a backend, but since Plastic SCM creates the databases itself, it's easier to specify the location here, in case you don't want to use the default.

Database suffix: the database suffix is a string appended to the name of every database created by Plastic SCM on the backend. This is useful if you plan to have several Plastic SCM servers using the same database backend, so they don't interfere with each other.

Before proceeding to the next step, test that everything is fine by click the "Test connection" button. If the test is completed successfully, the "Next" button is enabled and you can start with the migration.

Before starting, the wizard displays a reminder about stopping the server before doing the actual migration.

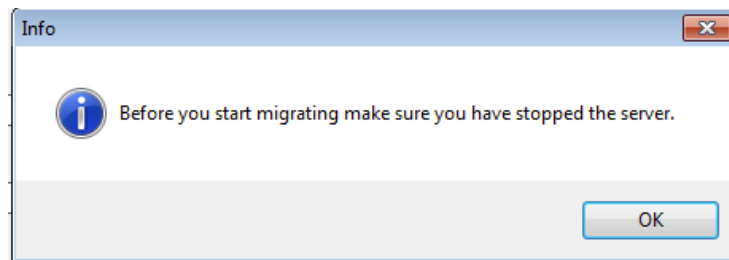


Figure 36: remember to stop the Plastic SCM server before migrating the database.

After clicking Ok, the wizard will move to the migration status page, indicating the overall progress of the operation and the progress of each individual repository. If everything is completed correctly, you will see the "migration succeeded" message at the end:

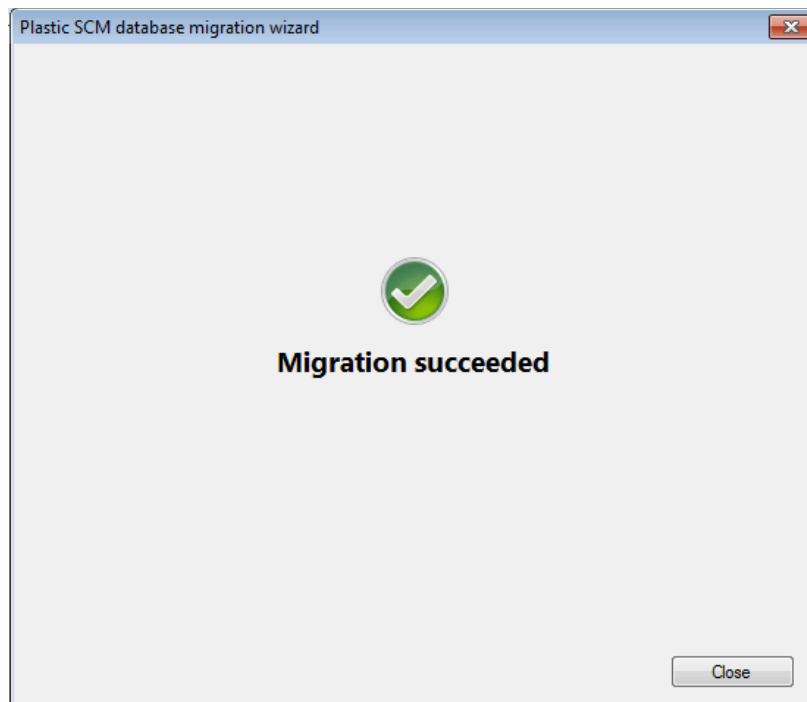


Figure 37: migration finished

When the database migration is completed, the wizard creates a new db.conf file with the database backend options specified in step 2 and renames the old one to db.conf.old. The next time the Plastic SCM service is started, it will connect to the migrated databases.

6.1.1 Oracle specific options

The Oracle database provider in Plastic SCM works in a different way compared to the others, due to its special way of managing databases. In the Oracle backend, Plastic SCM doesn't create a database for each repository. Instead, each repository is a *tablespace* inside the database specified by `oracle_sid` in the connection string. There are, indeed, two connection strings: the normal "*connection string*", used for connecting to each repository, and the "*admin connection string*", used to create the tablespaces. In this admin connection string, you have to fill in the following values:

- `_USER_`: user account to connect to the backend, with tablespace creation privilege.
- `_PASSWORD_`: password for the account.
- `_SERVER_`: the server machine with the Oracle backend.
- `_ORACLE_SID_`: the SID of the oracle database to connect to.
- `_MODE_`: normally "sysdba".

There is a textbox with the default tablespace creation commands. A working default is provided, but you may want to check with Oracle administrator on the right values for the different arguments since they define the grow rate and maximum sizes of the tablespaces.

7 Backup and restore

The backup and restore procedures are closely related to the database backend used in Plastic SCM. Out of the box, the Plastic SCM server uses an embedded SQL Server CE or embedded SQLite when running under Windows and Linux/MacOSX respectively.

The instructions in this chapter are meant only for the embedded backends. If you configured a different database backend, please check with your database administrator what are the best backup procedures for it.

7.1 How to backup the embedded databases

Backing up the embedded databases (SQL Server CE or SQLite) is just about copying the database files from disk. Each database will be a single file, so the operation is pretty simple. However, backing such a database file requires that the Plastic SCM server be stopped.

Note: This is one of the drawbacks of the embedded backends that is normally better handled by the other supported backends such as MySQL, MS SQL Server, Oracle or PostgreSQL. This is one of the reasons why we recommend using the embedded backends for evaluation purposes only.

Starting and stopping the Plastic SCM server on the command line has been described in the previous sections, so the following procedure can be easily automated with scripts.

Steps to backup:

- Stop Plastic server (`plasticd --stopservice`)
- Backup the database files from the server installation directory:

- The database files are repositories.plastic.* and rep_**.plastic.*.
- If running on Windows and using SQL Server CE (the default): backup all the files with ".sdf" extension.
- If running on Linux or Mac OS X and using SQLite, backup all the files with the ".sqlite" extension.
- Start Plastic server again (plasticd --startservice)

7.2 Restore embedded databases

The restore procedure is very similar to backup in reverse order:

- Stop Plastic server
- Copy all the files the backup to the server installation directory. If you want to restore only one repository, restore only the rep_xx.plastic.* file for that repository. The "repositories.plastic.*" file contains the list of repositories that are registered on the system, while the rep_xx.plastic.* files contain the data for each repository.
- Start Plastic server again.

8 Archiving revisions

8.1 Why archiving revisions

In a production environment where there are third party compiled tools or programs, binaries, big documents and other kind of big files that rarely change and / or are rarely accessed, it can consume disk space and time when storing those revisions in the database and afterwards retrieving them from the database.

To help minimize the impact mentioned above, you can use the archive command, which allows the administrator to set up a separate disk device, such as a tape, a USB pen drive, an external disk, a CD-ROM, DVD, or a specific disk space area, and store those big revisions there, so that they do not take space in the database. Thus, every time that a user needs to access those revisions, Plastic SCM will search for them in the external storage area, and retrieve that information to the user.

8.2 How to archive my revisions

To archive revisions, use the archive command:

```
cm archive C:\mybigfile.tar#br:/main#0 -c="big file of libraries"  
-f="/home/plastic/bigfileTARrev0"
```

This command will archive the revision 0 of the branch main of the item mybigfile.tar, creating several chunks of the revision; each one contains a part of the revision content. The comment of the archive is "big file of libraries" and the archive files will start with the prefix "/home/plastic/bigfileTARrev0". This means

that the archive will be created in that path. The `-f` parameter is a prefix for the archive files that can be used as a destination path for the archives. If the `-f` switch is omitted the archive files will be created in the directory of execution of the command.

It is possible to archive several revisions specifying them one after the other in the same command. To get further information about this command, type on a command line:

```
cm help archive
```

Once the archive files have been created the administrator can move them to the external data location. The next time that a user tries to access that data, Plastic SCM will try to get it from the external storage area.

Note: To create archived revisions it is mandatory that the user that executes the archive command is the repository owner. Otherwise the revision will not be archived correctly and Plastic SCM will continue using the database to access the data.

Warning: An archived revision cannot be archived again. Once a revision has been archived it is taken out the database. Be especially careful with the archived revisions or you will lose them definitely.

8.3 How are the archived revisions accessed

To access the data stored on an external location, a configuration file **externaldata.conf** file must be manually created. This file contains a path per line; those paths are the locations of the stored revisions. The following is a sample of an `externaldata.conf` file:

```
E:\archivesOfRepository1  
D:\mybigfiles\revisionsOfBigFileTAR  
F:\revisionsOfThe2_9Release
```

This file can be placed in two locations:

- On the server side: placing the **externaldata.conf** file on the Plastic SCM server location will allow every user of that server to access those revisions automatically, as long as the external storage area is available. This is the most useful option for system administrators.
- On the client side: placing the **externaldata.conf** file on the Plastic SCM client folder or on the user local directory (within Documents and Settings in Windows XP, Users in Vista/Seven, or home in UNIX based systems).

If a user tries to access to any stored revision from the GUI by executing an update, for example, and there is no **externaldata.conf**, a dialog will appear, asking for the location of the data as illustrated in the following picture:

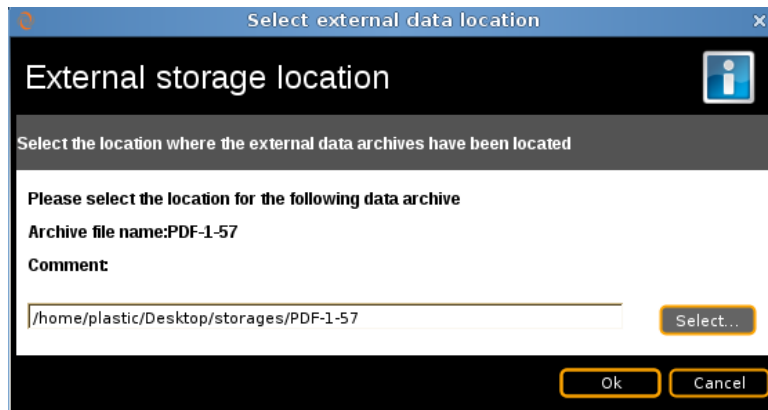


Figure 38. Introduce the external data location path

Once the first chunk of the revision is introduced, Plastic SCM will be able to find the other chunks of the revision, it will create an **externaldata.conf** file in the local user directory and from that moment on it will try to access to the archived revisions from that location. If Plastic SCM cannot access the data at a certain point of time, it will show the same dialog again, and if a new location is introduced, this location will be added to the existing **externaldata.conf** file.

From the CLI (command line interface) an **externaldata.conf** must be always available. Otherwise, the command will ask the user for an externaldata.conf to look for the revisions.

8.4 How to restore archived revisions

It is possible to save archived revisions back into the database, so that the archives can be safely deleted. From that moment on the database will be used to get the data. Example:

```
cm archive C:\mybigfile.tar#br:/main#0 --restore
```

This command will restore the revision 0 of the main branch of the file mybigfile.tar into the database, and the archives of that revision will not be used longer.

The external storage location must be available at the moment of the revision restoration, and an **externaldata.conf** must be available.

To get more information of this command, type on a command line:

```
cm help archive
```