

Plastic SCM 4.1

A guide for the Plastic SCM administrator

Plastic SCM **Administration**

© 2006-2012 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](#_Toc324420245)

[1.1 Plastic SCM 2](#_Toc324420246)

[1.2 Components 2](#_Toc324420247)

[2 Minimum requirements 4](#_Toc324420248)

[2.1 Server 4](#_Toc324420249)

[2.2 Client 5](#_Toc324420250)

[3 Plastic SCM installation 6](#_Toc324420251)

[3.1 Prerequisites 6](#_Toc324420252)

[3.2 Server and Client installation 6](#_Toc324420253)

[3.3 Server configuration 9](#_Toc324420254)

[3.4 User authentication configuration 11](#_Toc324420255)

[3.4.1 Authentication basics 12](#_Toc324420256)

[3.4.2 Local users 12](#_Toc324420257)

[3.4.3 Local users: name + ID 12](#_Toc324420258)

[3.4.4 Active Directory Integrated Security 13](#_Toc324420259)

[3.4.5 LDAP 13](#_Toc324420260)

[3.4.6 User/Password 13](#_Toc324420261)

[3.5 Server startup 19](#_Toc324420262)

[3.5.1 Windows Systems 19](#_Toc324420263)

[3.5.2 Linux Systems 20](#_Toc324420264)

[3.6 Client configuration 21](#_Toc324420265)

[3.6.1 Merge and Differences Tools configuration 23](#_Toc324420266)

[4 Using a proxy server 26](#_Toc324420267)

[4.1 Installing the Plastic SCM Proxy server 28](#_Toc324420268)

[4.2 Configuring the clients 29](#_Toc324420269)

[5 Creating and managing repositories 31](#_Toc324420270)

[5.1 Creating a new repository 31](#_Toc324420271)

[5.2 Listing available repositories 32](#_Toc324420272)

[5.3 Archiving repositories 33](#_Toc324420273)

[5.4 Reconnecting archived repositories 34](#_Toc324420274)

[6 Administration tool 36](#_Toc324420275)

[6.1 Summary 36](#_Toc324420276)

[6.2 Users 37](#_Toc324420277)

[6.2.1 Changing the authentication provider 38](#_Toc324420278)

[6.3 Database setup 41](#_Toc324420279)

[6.3.1 Connection configuration: specific backend details 43](#_Toc324420280)

[6.3.2 Common fields to all backends 44](#_Toc324420281)

[6.3.3 Migrating 44](#_Toc324420282)

[6.4 Listening ports 45](#_Toc324420283)

[6.5 Audit log 47](#_Toc324420284)

[6.5.1 Logging levels 49](#_Toc324420285)

[6.5.2 Validating the log 49](#_Toc324420286)

[7 Backup and restore 50](#_Toc324420287)

[7.1 How to backup the embedded databases 50](#_Toc324420288)

[7.2 Restore embedded databases 52](#_Toc324420289)

[8 Exclusive checkout configuration 53](#_Toc324420290)

[8.1 Configuring exclusive checkout in the server 53](#_Toc324420291)

[8.2 Exclusive checkout in a distributed environment 54](#_Toc324420292)

[9 Archiving revisions 56](#_Toc324420293)

[9.1 Why archiving revisions 56](#_Toc324420294)

[9.2 How to archive my revisions 56](#_Toc324420295)

[9.3 How are the archived revisions accessed 57](#_Toc324420296)

[9.4 How to restore archived revisions 58](#_Toc324420297)

**Figures**

[Figure 1. Installation directory 7](#_Toc324420298)

[Figure 2. Component selection 7](#_Toc324420299)

[Figure 3. Eclipse location 8](#_Toc324420300)

[Figure 4. Copying files 8](#_Toc324420301)

[Figure 5. Start system configuration 9](#_Toc324420302)

[Figure 6. Server configuration welcome screen 9](#_Toc324420303)

[Figure 7. Server configuration language selection 10](#_Toc324420304)

[Figure 8. Server configuration authentication modes 10](#_Toc324420305)

[Figure 9. LDAP Authentication configuration 10](#_Toc324420306)

[Figure 10. Port configuration 11](#_Toc324420307)

[Figure 11. Database backend setup 11](#_Toc324420308)

[Figure 12. User and password configuration screen 14](#_Toc324420309)

[Figure 13. Login dialog 14](#_Toc324420310)

[Figure 14. Configuration files in user/password mode 15](#_Toc324420311)

[Figure 15: umtoolgui usage 16](#_Toc324420312)

[Figure 16. users.conf file format 18](#_Toc324420313)

[Figure 17. groups.conf file format 18](#_Toc324420314)

[Figure 18. server.conf contains *WorkingMode* 18](#_Toc324420315)

[Figure 19. Managing Plastic SCM server service 20](#_Toc324420316)

[Figure 20. Client configuration welcome screen 21](#_Toc324420317)

[Figure 21. Client configuration language selection 21](#_Toc324420318)

[Figure 22. Workspace server selection 22](#_Toc324420319)

[Figure 23. Client authentication selection 22](#_Toc324420320)

[Figure 24. Client AD LDAP authentication window 22](#_Toc324420321)

[Figure 25. Example of proxy deployment 27](#_Toc324420322)

[Figure 26. Example of network 27](#_Toc324420323)

[Figure 27. Same network, improved by using two proxies 28](#_Toc324420324)

[Figure 28. Proxy installer binaries folder 28](#_Toc324420325)

[Figure 29. Cached data directory location 29](#_Toc324420326)

[Figure 30: Proxy listening port 29](#_Toc324420327)

[Figure 31: configuring a proxy server in the Plastic SCM client 30](#_Toc324420328)

[Figure 32: creating a new repository in the Plastic SCM GUI client 32](#_Toc324420329)

[Figure 33: New repository dialog 32](#_Toc324420330)

[Figure 34. Repositories view 33](#_Toc324420331)

[Figure 35: Administration tool summary page 37](#_Toc324420332)

[Figure 36: Users section 38](#_Toc324420333)

[Figure 37: authentication mode migration wizard 39](#_Toc324420334)

[Figure 38: LDAP configuration details 39](#_Toc324420335)

[Figure 39: User mapping in the old authentication mode and the mapping to the new. 40](#_Toc324420336)

[Figure 40: Editing a user mapping 41](#_Toc324420337)

[Figure 41: Database migration wizard 42](#_Toc324420338)

[Figure 42: database migration: target database options 43](#_Toc324420339)

[Figure 43: remember to stop the Plastic SCM server before migrating the database. 45](#_Toc324420340)

[Figure 44: migration finished 45](#_Toc324420341)

[Figure 45: Server listening port configuration 46](#_Toc324420342)

[Figure 46: certificate file selection dialog 47](#_Toc324420343)

[Figure 47: Enter certificate file password 47](#_Toc324420344)

[Figure 48; Audit log section 48](#_Toc324420345)

[Figure 49: Using an exclusive checkout master server 55](#_Toc324420346)

[Figure 50. Introduce the external data location path 58](#_Toc324420347)

**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](mailto:support@codicesoftware.com)

# Introduction

## 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 with other 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 and workspaces, and how to make backups.

## Components

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

* The server, which is responsible for storing all the project information and managing client access to that repository.
* The Clients, which 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 websiteto get a full and up-to-date list of compatibilities at [www.codicesoftware.com](file:///C:\Documents%20and%20Settings\Administrator\Desktop\ToDo_WORK\PlasticDocs_3.0\www.codicesoftware.com).

# Minimum requirements

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

## Server

The minimum system requirements to run the Plastic server are:

* Operating systems:
  + Windows XP SP2, Windows Vista, Windows 7
  + Windows Server 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 10.5 or higher
* .NET Framework 2.0 or higher if running on Windows.
* 512 MB RAM.
* Enough free hard disk space. Hard disk space will vary depending on the size of project’s assets.

## 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 XP SP2, Windows Vista, Windows 7
* Windows Server 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 10.5 or higher

On Windows, the .NET Framework 2.0 or higher is required.

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.

# Plastic SCM installation

## 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.

## 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 perform 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 check-in/check-out/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 Microsoft has deprecated it 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

## 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 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, specify 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 next 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 |
| The last step lets you optionally change the default database backend that stores Plastic SCM data.  Out of the box, Plastic SCM will use an embedded database that needs no configuration. However this database is meant for single user evaluation purposes. For production servers, we recommend that you configure a non-embedded database backend like SQL Server, MySQL, PostgreSQL, or Oracle.  For more details on the database setup, jump to section Administration tool. | Figure . Database backend setup |

Table 2. Server configuration

## 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 these allows different authentication possibilities and will be explained in the following sections.

### 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 based on different ways to build the SEID plus different ways to obtain users.

### 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 the user that’s logged on 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 an 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? | The server retrieves it from the local machine’s 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 username. |

### Local users: name + ID

The mechanism is identical to *Local users* but the SEID is built using both the username 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 username and ID.

|  |  |
| --- | --- |
| How does the server obtain the user list? | The server retrieves it from the local machine’s 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? | Username + ID: user id on Linux and SID on Windows. |

### 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? | The server retrieves it from the active directory main server. The server must be correctly configured. |
| How is the SEID built? | A Windows SID. |

### 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, this is useful 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. |

### 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 modes 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 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 into the Plastic SCM 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 in the following Figure 12.

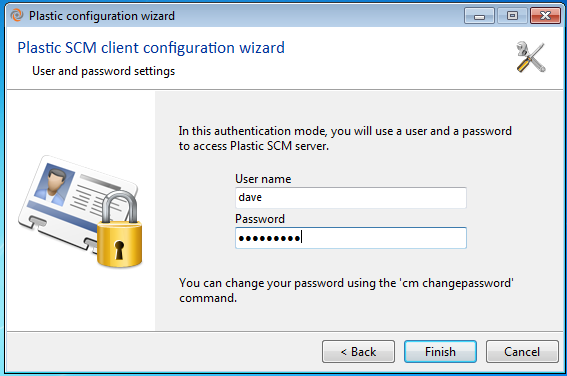


Figure 12. User and password configuration screen

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

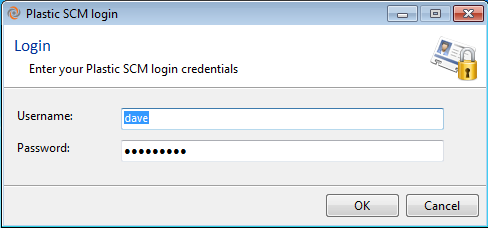


Figure 13. Login dialog

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 14. 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.
* The **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

#### 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 15 illustrates all the **umtoolgui** options. It is a simple and intuitive tool whose sole 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 15: 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.

**umtool** implements several commands detailed on the following Table.

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

Table 3 umtool 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.



Figure 16. users.conf file format

**groups.conf file format**

The groups.conf file contains all the groups known to the Plastic SCM 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. See the following figure for details.



Figure 17. 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.



Figure 18. 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 SCM server. This way Plastic SCM 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** (Graphical User Interface tool)

## 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.

### 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 19.

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

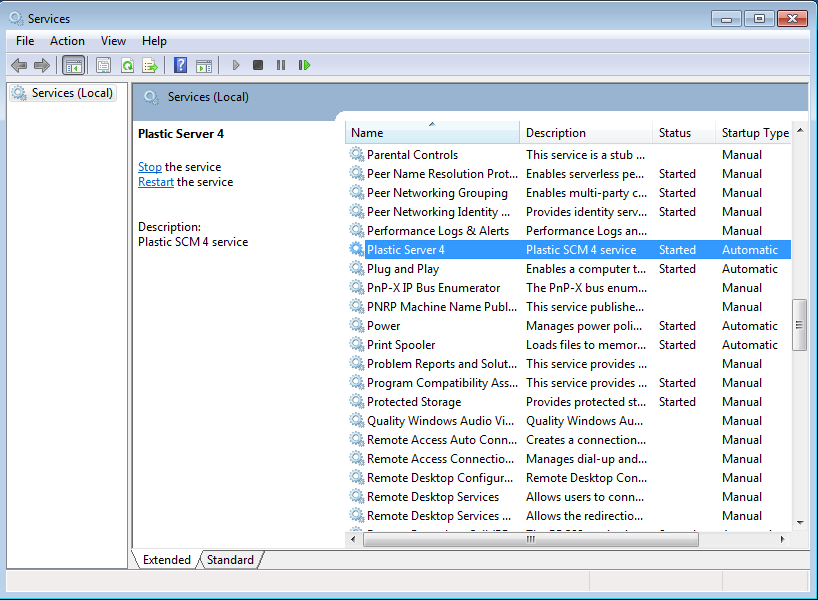


Figure 19. Managing Plastic SCM server service

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

### 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, the simplest way is to:

/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*.

## 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 20. Client configuration welcome screen |
| Language selection. | Figure 21. 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 .  By default, the Plastic SCM server TCP port is 8087. If it has been changed on the server, set the new port number here. | Figure 22. Workspace server selection |
| Next step is to choose an authentication mechanism as used in the Plastic SCM server. | Figure 23. Client authentication selection |
| If Active Directory integrated security was the Plastic SCM 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 24. 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.

plastic

### 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 you to specify 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 applies 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
  + @comparationmethod: 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 someone 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>

# 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.

Here are some 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 25. 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.

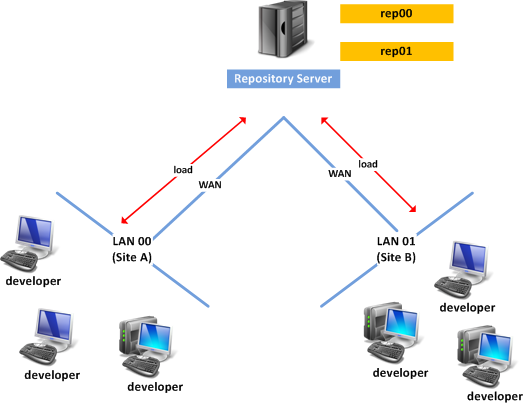


Figure 26. Example of network

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.

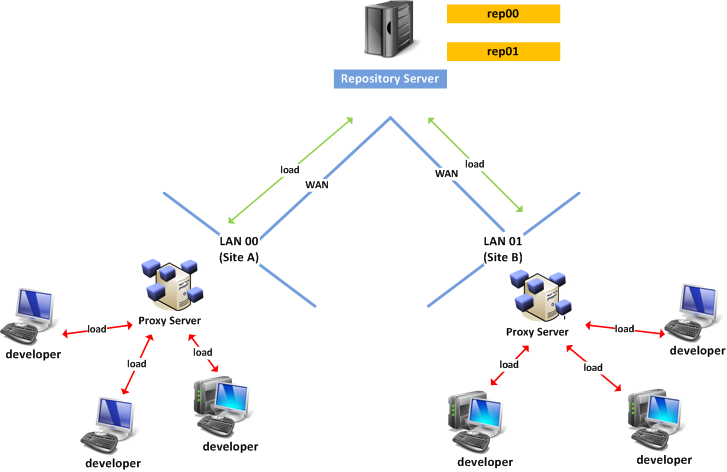


Figure 27. Same network, improved by using two proxies

## Installing the Plastic SCM Proxy server

The Proxy server is distributed as a separate download in the Plastic SCM website ([www.plasticscm.com](http://www.plasticscm.com)) and can be installed as a stand alone product: 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 28. 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 29. Cached data directory location |
| TCP Listening port.  This is the TCP port that the proxy server will use. By default, it is 8085. | Figure 30: 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. | |

## 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.

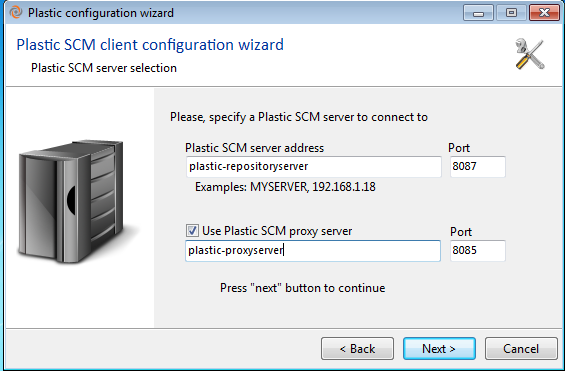


Figure 31: configuring a proxy server in the Plastic SCM client

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.

# 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.

## 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”.

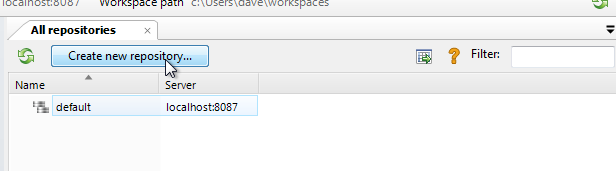


Figure 32: creating a new repository in the Plastic SCM GUI client

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

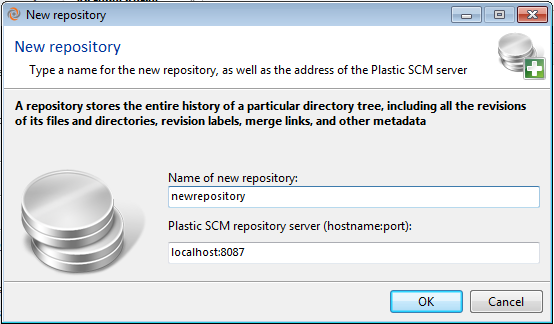


Figure 33: New repository dialog

## 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 34 shows the repositories view on the GUI tool, accessible from the *View* menu.

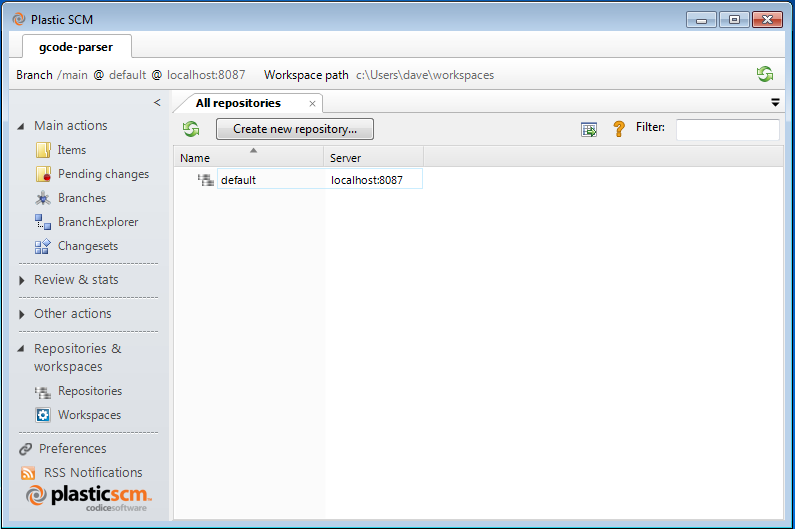


Figure 34. Repositories view

## 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.

1. It removes the repository from the list of registered repositories available to be used in workspaces.
2. 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.

## 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.

# Administration tool

The administration tool lets you configure several aspects of the Plastic SCM server. The tool has been designed to automate some tasks that used to be manual and error prone, like converting databases from one database backend to another, or changing the user authentication provider.

The Administration Tool is accessible from the startup menu in Windows, under the Plastic SCM 4 menu / Server. It is divided in sections, listed on the left panel. Clicking on a section displays its content in the panel on the right.

## Summary

The summary page displays an overview of the server configuration, covering version, license information, the authentication provider used to validate user logins and the database backend.

This is an informational page: it is not possible to set any values here. To do so, you will use the next sections.

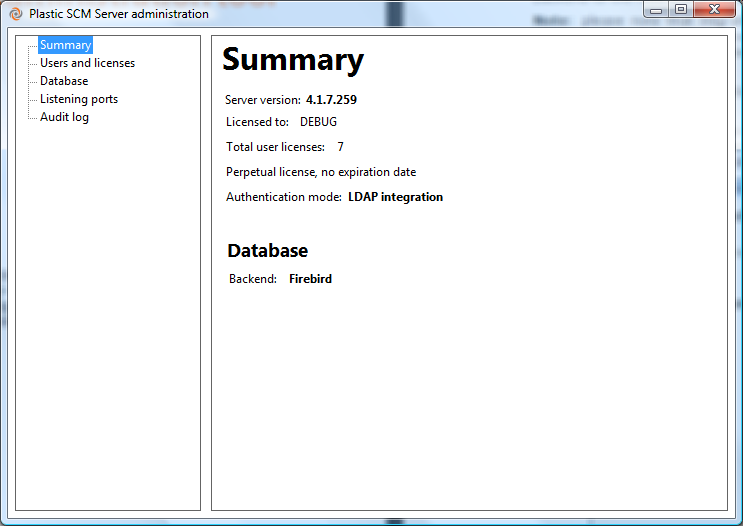


Figure : Administration tool summary page

## Users

The “Users” section displays detailed information about the installed license file (how many user licenses are available) and offers the ability to change the user authentication provider.

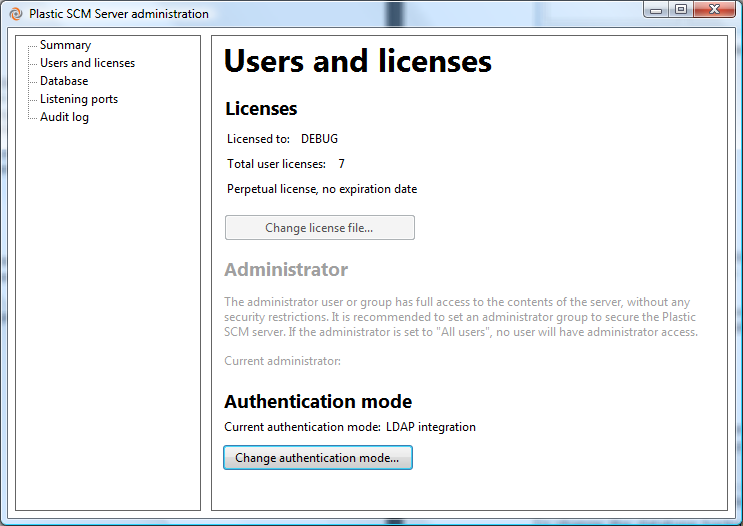


Figure : Users section

### Changing the authentication provider

When you configured the Plastic SCM server the first time, you were prompted to choose a way to authenticate your users, as described in section “” above. Maybe you choose to use the local users in the server machine through the “Name” authentication mode, and now you need to switch to Active Directory because your Plastic SCM deployment is growing.

This migration presents some technical challenges, since Plastic SCM needs to map the users in the old authentication mode to users in the new mode, and sometimes the names don’t match. Since Plastic SCM 4.1, a new wizard helps in this conversion, making the mapping mostly automatic for users with same names and greatly simplifying the process of matching the rest.

The migration wizard appears after clicking the “Change authentication mode…” button. The first step lets you choose the new authentication provider, as shown in Figure 37.

**Note: make sure to stop the Plastic SCM server during the migration.**

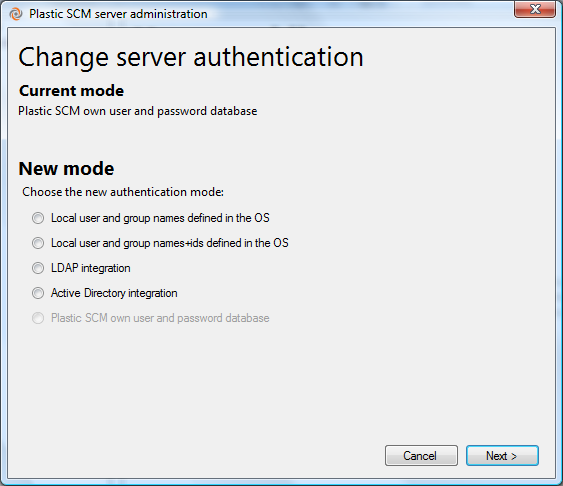


Figure : authentication mode migration wizard

For a detailed explanation of the different authentication modes, please refer to section “” above. If you choose “LDAP integration” as the target authentication mode, clicking the “Next” button will let you enter the details of your LDAP server:

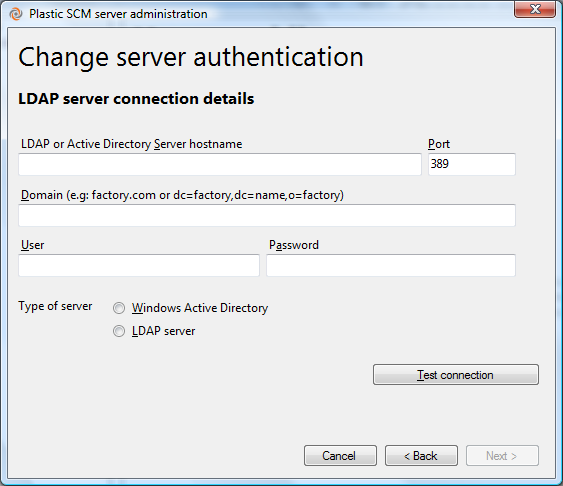


Figure : LDAP configuration details

The options in this step have already been described in section “”. Please refer to that section for specific details.

Clicking “Next” will load the list of users from the repositories databases in the current authentication mode and will try to map them to users in the new one. This mapping will simply try to match the names. For instance, if a user called “dave” existed in your User and Password database and you are migrating to Active Directory, a user called “dave” will be searched for there. If it exists, then the mapping is established.

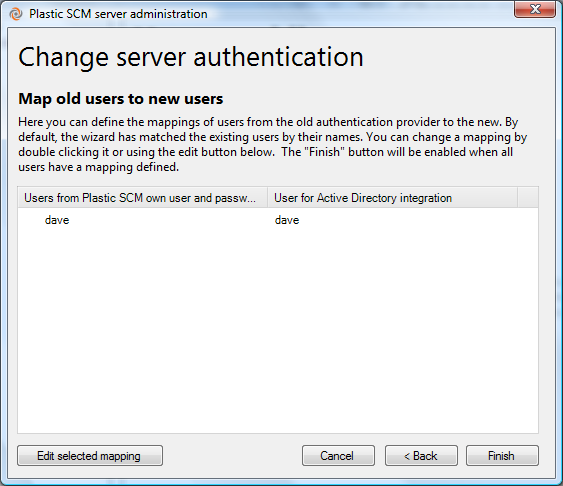


Figure : User mapping in the old authentication mode and the mapping to the new.

The left column displays the users from the current authentication mode. The right column displays the mapped user in the new system.

For those users not matched in the target authentication provider, the right column is empty. Those users need to be mapped manually. It is possible to edit the mapping for any users by clicking the “Edit selected mapping” button.

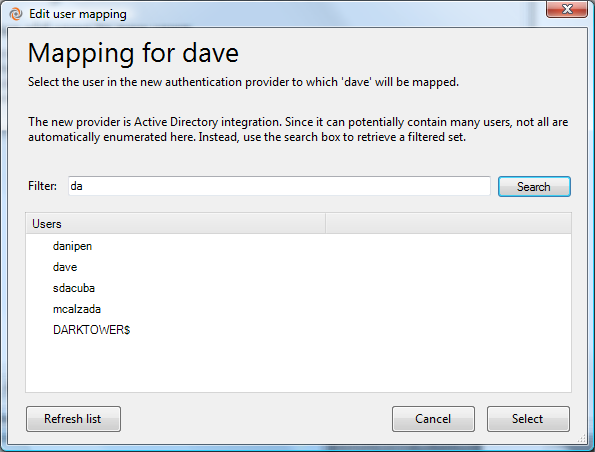


Figure : Editing a user mapping

This button lets you choose the user in the target authentication provider that maps the select user in the old provider. In Figure 40, for instance, the user “dave” from the old provider has been selected, and the “Edit user mapping” dialog is used to search for users containing “da”. In the results, you can select “dave” and the mapping will be set.

Once all users are mapped (either automatically or manually), the “Finish” button is enabled in Figure 39. Clicking it will perform the conversion. If the conversion is successful, a confirmation message appears.In case of error, any changes are rolled back, to ensure consistency.

Internally, the wizard will update all the user records (SEIDs) in the databases and once that is completed, it will also set the new authentication provider details in the file “server.conf”. The previous contents of the fil will be saved as “server.conf.old”.

## Database setup

Plastic SCM 4 supports 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.

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 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 41 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.

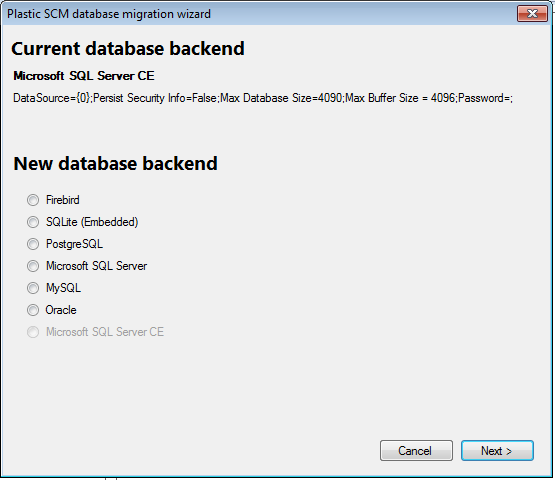


Figure 41: Database migration wizard

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

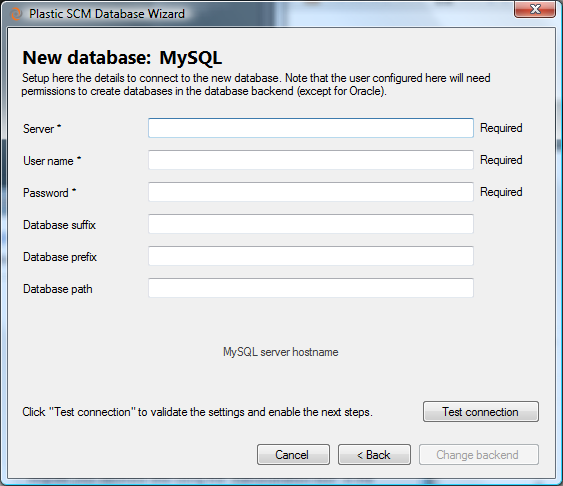


Figure 42: database migration: target database options

### Connection configuration: specific backend details

This step asks for the connection parameters needed to establish a connection to the database. Some options are specific for each database backends, while others are common to all.

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

The following table summarizes the specific fields for each backend:

|  |  |
| --- | --- |
| **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 | 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:  SERVER: the database server machine  ORACLED SID: the oracle\_sid of the database to connect to.  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.  MODE: normally “sysdba”. |
| MS SQL Server CE | By default, SQL Server CE uses the embedded backend, which doesn’t require any specific arguments. |

### Common fields to all backends

**Database path:** 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 prefix**: the database prefix is a string added in front of 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.

**Database suffix**: same as the prefix, but the content of this field is added to the end of the database name rather than the beginning.

### Migrating

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.

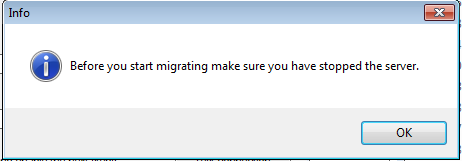


Figure 43: remember to stop the Plastic SCM server before migrating the database.

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:

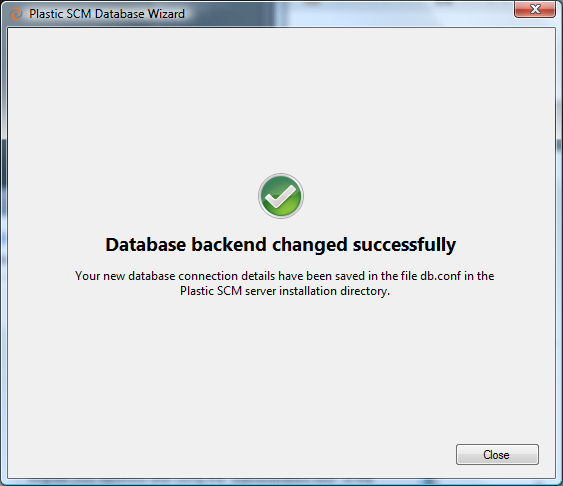


Figure 44: migration finished

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.

## Listening ports

The “Listening ports” section lets you configure the TCP ports where the Platic SCM server listens for incoming connections. It is possible to setup the encrypted (SSL) and non-encrypted ports, as well as the certificate used in SSL connections.

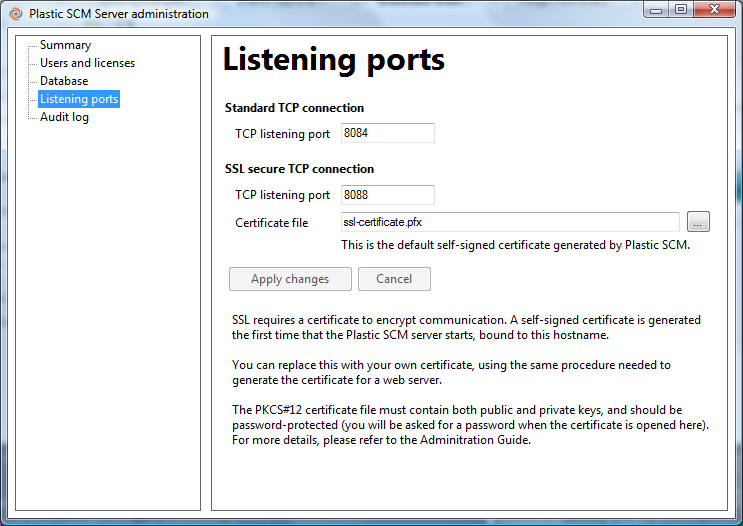


Figure : Server listening port configuration

To change any of the options, enter the new value in the textbox and select “Apply changes”.

For SSL connections, Plastic SCM uses a certificate file. On the first startup, the server will check the configuration for a certificate file and if none is configured, it will generate a self-signed certificate for the host it is running on. This file will placed in the server installation folder and named ssl-certificate.pfx.

The certificate contains private and public keys protected by a password in PKCS#12 format. This is the same type of certificate used for web servers, so if you want to use your own certificate signed by a trusted certificate authority instead of the self-signed one, you can do so just by generating a PKCS#12 file with your keys signed by the authority. These files are usually provided by the certificate authority itself for you and have a “.pfx” extension. They are normally protected by a password that you define. Ask your systems administrator for more details about getting a custom signed certificate for your server machine to be used in SSL connections.

To use the certificate file with your Plastic SCM server, click the “…” button and select the certificate file.

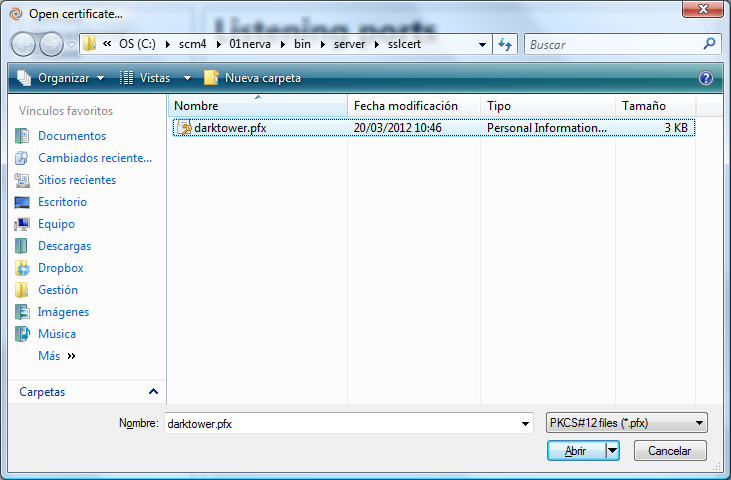


Figure : certificate file selection dialog

When you click “Apply changes”, you will be prompted for the certificate file password:

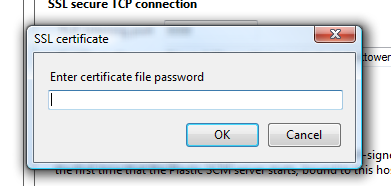


Figure : Enter certificate file password

The password is used to validate that the certificate contains the keys needed for SSL encryption and the configuration details are saved to the “remoting.conf” file.

A dialog reminding that you need to restart the Plastic SCM server for the changes to be effective is displayed.

## Audit log

Starting in Plastic SCM 4.1, the server generates a new type of log file that can be used for auditing purposes. This section configures what events are recorded (through the log level) as well as the file used for storing the log.

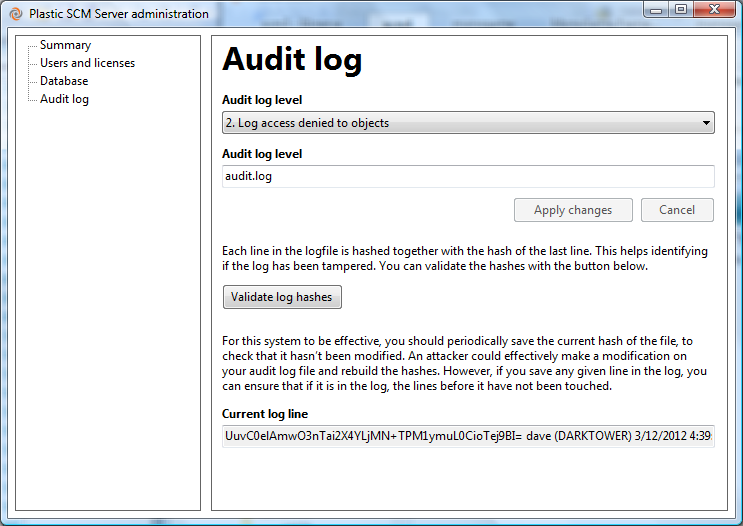


Figure ; Audit log section

The auditing log records operations in performed by the different clients on the Plastic SCM server. These operations are logged:

* Checkin
  + Added / changed / moved / deleted items
* Owner changed for any object
* Changeset deleted
* Object comment changed
* Add / rename / delete branch
* Add / rename / delete label
* Label reapplied
* Add / rename / delete attribute
* Attribute applied / removed from object
* Add / rename / delete repository
* Access denied to object
* Update (only revision id is logged)
* Annotate / diff (only revision id is logged)

For each logged operation, the following is recorded:

* Validation hash
* User that performed the operation
* Hostname of the machine from which the operation was performed.
* Timestamp of the operation
* Repository where the operation is performed
* Description of the operation (add, rename, delete) and the object that it affects. The object id os normally logged in parenthesis after the name.

Sample log line:

tRSMdT95z8H3MntMiljq/6rAfIdtsoB2THaAG9e9g6k= dave (DARKTOWER) 3/12/2012 4:38:47 PM [mainrep]: New label: BL00 (12) applied to changeset 0

### Logging levels

Since the audit log can quickly grow in size, different log levels are defined to limit the number of events recorded. This list details the levels available:

0: Nothing

1: Log deleted objects (including deleted items in a checkin)

2: Log access denied errors

3: Log New / Renamed / Edited objects

* Branches, markers, attributes, applied attributes, etc

4: Log read access to revisions. (Current status logs only the revision id)

By default, the log level is set to 1, meaning that it only logs objects that are deleted.

### Validating the log

Each line of output on the logfile is hashed together with the hash of the previous line. This helps identifying tampering of the log file. The “Validate log hashes” button validates the content of the log file to detect tampering using these hashes.

# 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.

## 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)

## 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.

# Exclusive checkout configuration

Certain types of files are not easy to merge when two users modify them in parallel. Examples of such files include images, compiled executable libraries, Excel documents or, in general, most binary files.

For these types of files, it is desirable to prevent several users making changes simultaneously, at least when working on the same branch. This is achieved in Plastic SCM using the *exclusive checkout* feature. With it, the administrator configures a list of file extensions so if any file with such an extension is checked out, that checkout will be exclusive; that is, no other user will be able to check it out until the initial user checks it in.

## Configuring exclusive checkout in the server

The configuration of the server is read from the file “lock.conf” located in the server installation directory. This file does not exist by default, you to configure exclusive checkout, you need to create it.

The file is made of rules that are applied to each repository. If a repository has no rules defined, no exclusive checkout happens for it.

* First line: the repository specification and lock server (more of the lock server later).
  + The repository in the format: “rep:*repositoryname*”
  + The lock server in the format: “lockserver:*server*:*port*”

Sample:

rep:default lockserver:myserver:8084

After the repository specification, each line contains a list of wildcards and file names. If a file matches the name or wildcard, a check-out operation on that file will be automatically exclusive (i.e: any other users won’t be able to check-out or check-in that file **in any branch** until the first check-out is undone or checked-in).

Sample filen rules:

\*.xls  
\*.xlsx  
document.vcs  
\*.pptx

Sample complete lock.conf file:

rep:default lockserver:myserver:8084  
\*.doc  
\*.xls  
\*.jpg  
document.vcs  
  
rep:myproject lockserver:myserver:8084  
\*.doc  
\*.docx  
\*.bin

## Exclusive checkout in a distributed environment

Plastic SCM is a distributed version control system (DVCS). In such a distributed environment, it can be difficult to perform exclusive checkouts: since exclusive checkouts are controlled in the Plastic SCM server and a distributed scenario may have many servers, some kind of arbitration is needed.

The solution is the *lockserver* parameter defined in the previous section. The idea is to define one of the distributed servers as the “*exclusive checkout master*”, so that all checkout operations will check it to determine if a checkout can be performed or not.

Any Plastic SCM server can act as a *checkout master*, so you can define as many as needed. For instance, if you have several offices with distributed developers in them, each having his own server, you may want to setup a lock.conf file on each developer server pointing to the office server. However, you want to have a checkout master on each office, since the connection between offices may not be fast or reliable.



Figure : Using an exclusive checkout master server

The master server will just keep a list of item GUIDs that are checked out exclusively and will not let any item with that GUID to be checked out when asked. Keeping a list of GUIDs means that the item is correctly identified even if it has been renamed differently on several replicated repositories. The only limitation is that the servers need to have an active network connection.

# Archiving revisions

## 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.

## 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.

## 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:

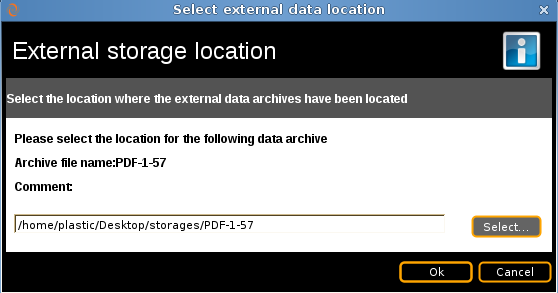


Figure 50. Introduce the external data location path

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.

## 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