



CM
bridge
Admin Guide

- 1 Introduction
 - 1.1 Overview
- 2 License
 - 2.1 Evaluation License
 - 2.2 Full License
- 3 Installation
 - 3.1 Prerequisites
 - 3.1.1 CM-Bridge Server User Account (on Linux)
 - 3.1.1.1 User Id (and Group)
 - 3.1.1.2 PATH considerations
 - 3.2 CM-Bridge Installation
 - 3.3 Upgrading from CM-Bridge 2.x to CM-Bridge 3.x
 - 3.3.1 Database Backup and Upgrade
 - 3.3.2 Config File Backup and Upgrade
 - 3.4 Upgrading from CM-Bridge 3.x to 3.x or above
 - 3.4.1 Database Backup and Upgrade
 - 3.4.2 Config File Backup and Upgrade
- 4 Configuration
 - 4.1 Subversion Branching on Update
 - 4.1.1 Branch Format
 - 4.1.2 Branch Location
 - 4.1.3 Merge Log Messages
 - 4.2 Logging
- 5 Planning Bridges
 - 5.1 Special Characters in Git file names
 - 5.2 Setting up a New Bridge
 - 5.2.1 Recommended Initial Bridging Method
 - 5.2.2 Bridging Two Repositories With Existing Data
 - 5.2.3 Defining Your Bridges
 - 5.2.4 Optional Initial Synchronization
 - 5.3 Suspended Bridges
 - 5.4 Disabled Bridges
 - 5.5 Subversion Hooks
- 6 ClearCase to Subversion/Git/Mercurial Bridges
 - 6.1 Functionality
 - 6.2 Considerations
 - 6.2.1 General
 - 6.2.2 ClearCase
 - 6.2.2.1 ClearCase Conflicts
 - 6.2.3 Subversion (for ClearCase to Subversion Bridges)
 - 6.2.4 Git (for ClearCase to Git Bridges)
 - 6.2.5 Mercurial (for ClearCase to Mercurial Bridges)
 - 6.3 Multiple Bridges for the Same Repository
 - 6.4 Constraints and Limitations
 - 6.4.1 General
 - 6.4.2 ClearCase
 - 6.4.2.1 ClearCase Attribute
 - 6.4.2.1.1 Automatic creation of ClearCase Attribute
 - 6.4.3 Subversion (for CC2SVN bridges)
- 7 Subversion to Subversion Bridges
 - 7.1 Functionality
 - 7.2 Considerations
 - 7.3 Multiple Bridges for the Same Repository
 - 7.4 Subversion Properties
 - 7.4.1 Special Properties
 - 7.5 Constraints and Limitations
- 8 ClearCase to ClearCase bridges (Via Subversion)
- 9 Multi-Site Bridges
 - 9.1 Star Setup
 - 9.1.1 Example Bridge Set up
 - 9.1.1.1 Notes
 - 9.2 Chain Design
 - 9.2.1 Example Bridge Set up
 - 9.2.1.1 Notes
 - 9.3 Merge Conflicts
- 10 Bridge Webadmin Configuration
 - 10.1 CM-Bridge Webadmin Server
 - 10.1.1 Additional Users and Permissions
 - 10.1.2 Manual Bridge Synchronization
- 11 CM-Bridge Server
- 12 Dealing with Merge Conflicts and Other Error Notifications
- 13 Troubleshooting
 - 13.1 Workspace Clean-up
 - 13.2 Forcing a Resynchronisation

- 13.3 Database Access Fails
- 14 Appendix A: CC2Git on Windows
 - 14.1 Prerequisites
 - 14.2 Steps
 - 14.2.1 Server or remote repository
 - 14.2.1.1 Setup
 - 14.2.1.2 Create Repo
 - 14.2.2 Client or local repository
 - 14.2.2.1 Setup
- 15 Appendix B: Setting up SSH Keys for automatic login to Git SSH repositories
 - 15.1 Prerequisites
 - 15.1.1 Linux
 - 15.1.2 Windows
 - 15.2 Setup
 - 15.2.1 Linux
 - 15.2.2 Windows
 - 15.2.2.1 msygit
 - 15.2.2.2 Cygwin Git

Introduction

Welcome to Clearvision's CM-Bridge : Integrating Subversion, Git, Mercurial and ClearCase.

The CM-Bridge is a multi functional bridge, which lets users of Subversion, Mercurial and Git carry on using their repositories, while at the same time automatically sharing code with ClearCase teams who are using ClearCase tools. Changes made at *either side* are seamlessly transported across the bridge to the other side. This way teams can collaborate across version control systems allowing Enterprise centralization and control while allowing localized freedom for other parts of the team.

From version 2.0 onwards, the CM-Bridge supports Subversion to Subversion Bridges. This **new** functionality provides the benefits of a Subversion multi site without the need to install separate applications.

This can be used as a permanent 2-way bridge, or to support a strategy of slowly migrating to another SCM System.

The CM-Bridge (also known as CC2X) implements CC2SVN, SVN2SVN, CC2HG and CC2GIT functionality. The distribution package and executables are named using the `cmbridge` prefix. Note that there is only one distribution for the CM-Bridge that contains these products. You can use the same installation to create Git, Mercurial and Subversion bridges.

Overview

The CM-Bridge provides a bridging mechanism that allows you to create a transparent link between your repository on one side of the Bridge and your repository on the other side.

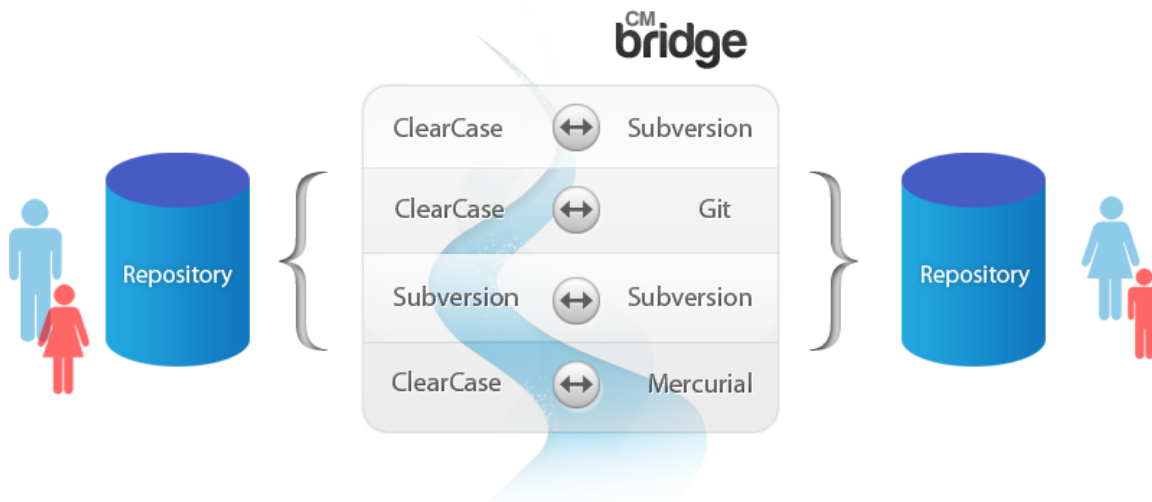
The currently supported configurations are:

- Subversion to Subversion
- ClearCase to Subversion
- ClearCase to Git
- ClearCase to Mercurial

Data checked in to either end of a bridge will automatically be transferred to the opposite end. The CM-Bridge operates asynchronously, meaning that when changes are committed in Subversion, Mercurial or Git or checked in to ClearCase, they are transferred in the background to the associated SCM system. The CM-Bridge checks each bridged repository location on a regular basis to scan for new changes, which are then transferred to the other end of the bridge automatically.

Linking between SCM systems is achieved through bridges. Internally, each bridge maintains two workspaces: **e.g.** a ClearCase workspace (i.e. a view) and a Subversion workspace. The bridge synchronizes between the two workspaces and deals with updating and committing changes to and from these workspaces.

Note: The common focus between all bridges is the transferring of the main content that is held in files. This ensures that a consistent set of files is available at either end of the bridge. The exact nature of the data and meta-data transferred between each side of the bridge is specific to each bridge type and will be outlined below in the relevant sections.



License

If no Evaluation or Full license has been requested, CM-Bridge will run in 'Restricted Mode'. This will allow you to use CM-Bridge with full functionality but only for a limited time (5 Bridges, 20 updates).

Evaluation License

Evaluation licenses can be requested from the clearvision website: <http://www.clearvision-cm.com>

A link to the license request form for a product can be found on the product's download page. To request a license we need some information about the machine that you will be running the server on. To get this information, you can run the `get_license_info` program on the machine you want to use as the server (the `get_license_info` program can be found within the tools directory of the installation directory). This will retrieve and present system information from the machine.

Once the license request form has been completed and submitted, an email will be sent to the registered email address containing the license information. This information will need to be inserted into the server config file.

Note that this license will only work on a particular server machine, on a particular port, and will allow a maximum number of users. The license is time-limited so it is recommended that Evaluation licenses are not requested until needed.

Full License

Once a license has been purchased, you will receive an email containing a link to the online full license generator. A similar form will need to be filled in as for an evaluation license. Once the form has been completed and confirmed a full license will be emailed. Any evaluation license details in the config file will need to be replaced by the full license.

Licenses are tied to the hardware as well as the IP address of the machine used for the server. It is therefore necessary to ensure these details will not change as this will leave the license invalid.

```
License Request for Clearvision CM-Bridge:
  Evaluation: yes/no
  IP Address: 192.168.64.1
    Port: 54329
  MAC Address: 00-50-56-C0-00-05
    Users: 50
  Company: You Company name here
  Email Address: yourco@yourco.com
  Expiry Date: 2010-03-31
  Product: CC2X
  Num of Bridges: 5
```

The `get_license_info` executable is run from a command prompt, e.g. on Windows:

```
Please either cut and paste the form and email the request to
sales@clearvision-cm.com or visit our website and fill in the
form online at: http://www.clearvision-cm.com/products/

Note: ALL fields are required

-----
Server License Request:
Company Name=_____
Contact Email Address=_____
Required Expiry Date=_____ (eg 2008-12-31)
Server port number=_____ (eg 54321 for CQ2SUN, 54322 for Jira2SUN)
Number of Users=_____

PRODUCT= CQ2SUN or JIRA2SUN
HOSTNAME=tw-cc2svndemo

:      IP=192.168.66.131      MAC=00-0C-29-C9-8D-7C
-----
```

Installation

To install the CM-Bridge products, you will need to do the following on your Server machine:

- Take care of prerequisites (user ids, basic configuration of ClearCase, Subversion, Mercurial and Git)
- Install the CM-Bridge (which has 2 parts: the main Server and the administration GUI Server)

Prerequisites

The following prerequisites need to be in place before you can start using the CM-Bridge

CM-Bridge Server User Account (on Linux)

User Id (and Group)

If you are running on Linux, we recommend setting up a separate user (e.g. `cmbridge`) to run the CM-Bridge Server and contain the workspace and config files, etc. This user can then be used to run the CM-Bridge server process and you can control this user's access to ClearCase, Subversion, Mercurial and Git appropriately.

Note: Please make sure that you set the human-readable name of the user (i.e. the 5th field in `/etc/passwd`, see the `chfn` command) otherwise you may have problems with Git, Mercurial.

PATH considerations

On Linux, make sure that ClearCase, Subversion, Mercurial and Git client software is usable by the CM-Bridge User Id.

CM-Bridge Installation

Start by unpacking the zip file on the machine to be used as the CM-Bridge server host. Note that files can be extracted in any location. However, recommended locations are `C:\Program Files` on Windows and `/opt` on Linux hosts. The following directory structure will be created:

- `/opt/Clearvision/cmbridge/` (or `C:\Program Files\Clearvision\cmbridge` on Windows by default)
 - `admin`
 - `docs`
 - `server`
 - `tools`
- On Linux, adjust permissions using `chmod -R 755 Clearvision` or `chmod -R a+x Clearvision`

Note: It is recommended to install and run the CM-Bridge server and the Webadmin server under the same user id. If you do need to use different user id's, then please make sure file permissions are set appropriately on the workspace directory, in particular the `webadmin` subdirectory and the files it contains.

Upgrading from CM-Bridge 2.x to CM-Bridge 3.x

Users running CM-Bridge v2.0 and above who are looking to upgrade to v3.0 will need to carry out a few additional tasks to make sure that their current bridges will not be lost. Follow the steps below to make sure a successful upgrade occurs:

Database Backup and Upgrade

- Stop your CM-Bridge Server and CM-Bridge Web Server
- **MAKE A BACKUP OF YOUR CURRENT DATABASE FILE** (`cc2x.db`, located in `<cm-bridge_installation_folder>/admin` by default)
- After these processes have been successfully terminated, copy your existing `cc2x.db` file into your new CM-Bridge installation
- The final step is to rename the `cc2x.db` database file to `cmbridge.db`

Config File Backup and Upgrade

- Stop your CM-Bridge Server and CM-Bridge Web Server
- **MAKE A BACKUP OF YOUR CURRENT CONFIG FILE** (`cc2x_server.cfg`, located in `<cm-bridge_installation_folder>/server` by default)
- After these has been successfully terminated, copy your existing `cc2x_server.cfg` config file into your new CM-Bridge installation
- The final step is to rename the `cc2x_server.cfg` config file to `cmbridge_server.cfg`

Following these steps will successfully upgrade your CM-Bridge installation without the need to recreate any of your current bridges. After these steps have been completed, you can archive your old `cc2x` installation and continue CM-Bridge with the new installation.

Upgrading from CM-Bridge 3.x to 3.x or above

Users running CM-Bridge v3.0 or higher who are looking to upgrade to v3.x or above will need to carry out a few additional tasks to make sure that their current bridges will not be lost. Follow the steps below to make sure a successful upgrade occurs:

Database Backup and Upgrade


- Stop your CM-Bridge Server and CM-Bridge Web Server
- **MAKE A BACKUP OF YOUR CURRENT DATABASE FILE** (cmbridge.db, located in <cm-bridge_installation_folder>/admin by default)
- After these processes have been successfully terminated, copy your existing cmbridge.db file into your new CM-Bridge installation
- Clearvision recommend keeping the backup of your old cmbridge.db until you are satisfied that all bridges are working as expected in your new installation

Config File Backup and Upgrade

- Stop your CM-Bridge Server and CM-Bridge Web Server
- **MAKE A BACKUP OF YOUR CURRENT CONFIG FILE** (cmbridge_server.cfg, located in <cm-bridge_installation_folder>/server by default)
- After these processes have been successfully terminated, open up the new cmbridge_server.cfg (which is part of the CM-Bridge download)
- Copy the *[LICENSE]* section from your recently backed-up cmbridge_server.cfg and copy it into the brand new cmbridge_server.cfg. This will preserve your license and allow you to run your new installation of CM-Bridge.
 - You cannot simply copy the old file and reuse it within your new installation as additional configuration options have been added

Following these steps will successfully upgrade your CM-Bridge installation without the need to recreate any of your current bridges. After these steps have been completed, you can archive your old CM-Bridge installation and continue CM-Bridge with the new installation.

Configuration

 Please take care when editing any configuration files. Text editors such as Vim, Wordpad or Emacs can be used for editing. However, avoid Notepad or MS Word as extra control characters may be introduced, causing the application to fail.

Copy the template config file `/opt/Clearvision/cc2svn/server/cmbridge_server.cfg` to an appropriate place, e.g. on Linux you might use `/home/cmbridge/.cmbridge.cfg`, on Windows, `c:\cmbridge\cmbridge.cfg`

Edit it to configure the following parameters:

(Note: Avoid Notepad or MS Word as these may introduce unwanted characters, corrupting config files)

```

[General]
; Working area for the {nl:ClearCase} Bridge
workspace=/opt/Clearvision/cmbridge/workspace

; The location the product is installed into (EG. /opt/Clearvision/cmbridge)
;install_location=

; The amount of delay between each bridge sync
update_delay=60

; Email server information to allow the CM-Bridge to send email notifications on problems
smtp_host=mail.server.com
smtp_username=user
smtp_password=password
from_address=user@localhost
; Default recipient for email notifications. Can be overridden in webadmin interface
default_resolvers=user@localhost

[Options]
; This section outlines additional options

; The svn_ignore_file_props and svn_ignore_rev_props dictate which properties should not be
transferred between Subversion
; repositories on a Subversion to Subversion Bridge
; svn:mergeinfo should not be removed from svn_ignore_file_props
svn_ignore_file_props=svn:mergeinfo,clearvision-cc2x
svn_ignore_rev_props=myprop,yourprop,hello:myprop

; The svn_cm_bridge_branch tells CM-Bridge where to store the branches it creates
; for each bridge when performing a bridge transfer
svn_cm_bridge_branch=branches


[License]
company_name=Your Company Plc
licence_email=admin@yourcompany.com
expiry_date=2008-12-31
bridges=999
license_key=PLEASE_ADD_YOUR_LICENCE_key_HERE
license_certificate=PLEASE_ADD_YOUR_LICENCE_certificate_HERE


[Server]
; Define this {nl:ClearCase} Bridge server machine as per your license
server_ip=192.168.1.2

[WebAdmin]
; Port for webadmin server
webadmin_port=8181

```

The License and Server sections must match the licence received from Clearvision. Any changes to the License or Server section will invalidate your licence.

 The workspace defined in the config file **must** point to a local file system and not a network drive. Also, on older Windows systems, if the workspace lives on a FAT file system, you will need `share.exe` running (for file locking to work correctly).

 If you do not correctly specify the **install_location**, you may receive an error message when running the `cmbridge_webadmin` application. CM-Bridge will attempt to calculate what the `install_location` should be but we recommend that this is set before starting the tool (EG. `C:\Program Files\Clearvision\CM-Bridge\cmbridge`)

Subversion Branching on Update

CM-Bridge V2.1 introduces new functionality to avoid multiple conflicts during updates and to further safeguard your Subversion repository during updates.

In order to do this, CM-Bridge will create a branch and copy all the changes from the other side of the bridge into this branch.

After completing the bridge update for the side, CM-Bridge will merge the branch into the main path (specified in the **CM-Bridge Webadmin**

Server).

This has a number of benefits:

1. If any issue occurs during a bridge update, all the data transferred is left on the branch
 - The main line is unaffected, meaning you have time to correct any errors and re-run the update
 - This avoids 'partial' bridge updates being merged into your main line, which could cause mixed code versions
 - On the next bridge update, the bridge will resume updating from the last checkpoint (essentially exactly where it started on the previous update)
2. Merging is much more simple and complicated merge files are reduced

Note: ;CM-Bridge creates a branch directory by default at the root of the repository.

It is possible to alter this functionality within the CM-Bridge Server configuration file. However, it should be noted that the branch location should be ABOVE the level of the Bridge location in the repository. The reason for this is to avoid creating large and complex branching structure which goes against best practices within Subversion.

Branch Format

In order to perform these updates, CM-Bridge will create a branching directory called **.clearvision_cmbridge** in the root of the repository. Each Bridge update will then create a unique sub-directory in the branch directory, with the following format:


- **<bridge_name>_YYYYMMDDHHMMSS**

This format allows you to see what bridge created the branch and at what date and time it was created.

Branch Location

 The branch location applies to **all Subversion repositories** managed by CM-Bridge, therefore it is important that this is set up correctly.

Although the default location of the update branch is the root of the repository, it can be altered.

- Specify an alternative branching directory using the **svn_cm_bridge_branch** option in the config file (under the **Options** section)
-  It is not necessary for this directory to exist as CM-Bridge will create it for you. Be aware of this fact as it will create the entire folder structure specified by **svn_cm_bridge_branch** option (in addition it will create the **.clearvision_cmbridge** folder)

Merge Log Messages

In order to allow you to see where a commit came from a merged branch, CM-Bridge will write the relative path of the branch into the log file (at the end of the log messages)

Logging

The CM-Bridge keeps log files for both the CM-Bridge server and the Webadmin server. You can configure the location of the log files using the **log_file** and **web_log_file** options, and the level of detail using the **log_level** option. The level can be one of the following:

- **INFO:** This provides verbose output about the operations of the CM-Bridge, which is particularly useful when investigating issues. Please bear in mind that due to the amount of output generated, the log file will grow quickly on this setting. **INFO** is a good log level to choose when you first install the product and configure bridges as it will give you the opportunity to review progress.
- **WARNING:** (*the default*) This shows warning and error messages, including details of possible issues. You may want to operate on **WARNING** level for a few days until you are satisfied that the product has been configured correctly, and switch to **ERROR** when you are happy.
- **ERROR:** This shows only error messages (minimal output), keeping the log file small.

For example:

```
log_file=/var/log/cmbridge/cmbridge_server.log
web_log_file=/var/log/cmbridge/cmbridge_web.log
log_level=INFO
```

Planning Bridges

Before setting up bridges, please read through this guide carefully to allow you to make an informed decision about your bridge setup. You can

save time (and effort) by planning your bridges well and creating a suitable setup.

If you later discover that a setup was not suitable for your needs, you may need to perform manual modifications in one or more of the repositories in order to rectify the situation. Time spent on planning bridges up-front can save you time in the long run.

Special Characters in Git file names

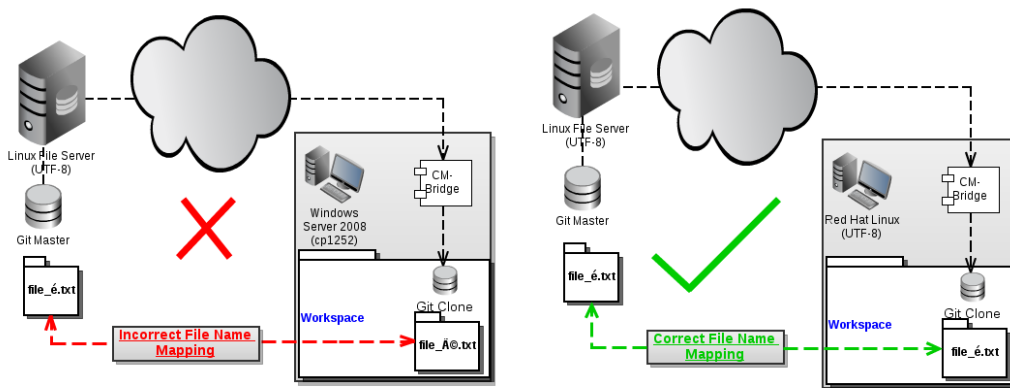
Special Characters are generally viewed as characters that fall outside the standard ASCII character set (e.g. French or German characters that have accents).

In Order to allow Special Character Support for Git Bridges, special consideration is needed.

1. All Git repositories must be hosted on the same file system type
2. The CM-Bridge application must also reside on the same file system type as **all** the Git (master) repositories

This requirement is necessary because of the way that Git stores special characters. See the following Git FAQ (https://git.wiki.kernel.org/index.php/GitFaq#Does_git_convert_encodings_of_file_names.3F) for confirmation.

The following Diagram shows a correct and incorrect setup of git repositories.



It is not always possible to know what file system the Git master repository is saved on (especially when accessing a git repository using https:// or git:// protocols over a network); therefore CM-Bridge has no way of knowing the source encoding for the Git repository.

In the above left diagram, the git repository is located on a Linux file system. A Linux file system is UTF-8 encoded therefore when transferred onto a Windows file system, the file name show incorrectly due to the fact that Git stores file names as byte sequences. This will result in the Bridge creating files at the ClearCase side of the bridge with what looks like the wrong file name.

In the above right hand diagram, the bridge will operate correctly as both the Git master repository and the CM-Bridge application sit on the same file system type (i.e. both UTF-8 based file systems).

This situation is important only where special characters are being used.

Setting up a New Bridge

Recommended Initial Bridging Method

When you first set up a new bridge, it is strongly recommended that one of the repositories is used as the 'master' for the initial synchronization. When Bridging from ClearCase to Git/Mercurial or Subversion the 'master' is usually the ClearCase side of the Bridge.

In this set-up, only the 'master' side contains data in the location specified for the bridge. The 'other' side points to:

1. An empty directory in Subversion
2. An bare/empty repository/branch in GIT or Mercurial.

CM-Bridge will detect all data that needs to be transferred.

Bridging Two Repositories With Existing Data

1. If you were to bridge between two repository locations that already contain data, you would effectively merge the two locations together,

which can result in conflicts.

- You would need to resolve these manually.



If you do need to merge two such repositories together, it is recommended that you

1. Perform this merge manually on one side.
2. Make this side of the Bridge (i.e. the manually merged data) the master for the initial transfer.

This will reduce the risk of the operation not having the desired result, and it will make it easier to undo any undesired results.

Defining Your Bridges

When you define a bridge, please note that both of the ends of the bridge must exist, i.e. if you set up a bridge from <http://host.domain.com/svn/myrepository/trunk/mycomponent> (on the Subversion side) to `M:\my_view\myproject\mycomponent` (on the ClearCase side), then Subversion must contain a directory called `mycomponent` at the specified location, and the ClearCase element `mycomponent` must also exist and be accessible via the specified path. In terms of masters for the initial transfer, if Subversion is the master for this bridge, then CM-Bridge would expect to find data for `mycomponent` in the Subversion repository and an empty directory `mycomponent` in ClearCase. Any sub-directories will be copied over the bridge as part of the initial transfer so do not need to be manually created.

Note: Once the initial transfer has been performed, neither of the two ends of the bridges retain any particular ownership (or master relationship). This means that you can then edit changes in either location and they will be transferred to the other end of the bridge automatically.

Optional Initial Synchronization

New in version 3 of CM-Bridge, new Bridges have the option to *not* perform an initial synchronization between both sides. When creating your Bridge, you now have the option to start it and only transfer data which is added *after* the creation. Before this, all data would be transferred across and each Bridge would be completely in sync after the initial update has completed. Now, if both sides contain different data and you only want to synchronize the bridge from the moment of the creation (and therefore only files added after the Bridge has been created), this is now possible. The default value is to perform an initial synchronization.

Suspended Bridges

CM-Bridge has the ability to put the bridge into a Suspended state. A bridge is put into a suspended state when an error (such as a conflict within the workspace) occurs and requires the users' input to solve. This means that the CM-Bridge server will not need to auto-shutdown if such an error occurs and can carry on synchronizing other bridges if they exist and are not in a suspended state.

When a Bridge is in a suspended state, CM-Bridge will provide output within the log file and command prompt/terminal explaining how to solve the issue. Using the example of a merge conflict, a user will need to go into the workspace and manually solve the merge conflict error and then remove a text file which CM-Bridge creates to know which Bridges are in a suspended state. After the merge conflict has been fixed, the text file will need to be deleted (with the location being displayed within the log file). After this has been completed, the Bridge will come out of a suspended state and synchronization will resume when the next auto/manual update is scheduled.

Disabled Bridges

As of CM-Bridge v3.1, Bridges can be enabled and disabled in real time. This is performed via the CM-Bridge Web Interface, as seen in the screen-shot below ('active' column).

Subversion to Subversion Bridges

Change	Id	Name	Active	Initial Sync	Resolver Emails	Side	Repository	User Name	Password	Delete	Synchronize Bridge
<input type="checkbox"/>	1	cmbidge_bridge_1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	cmbidge@no-reply	Subversion	file://home/andy/Documents/CV_WORK/SVN_REPOS/rep01				<input type="button" value="Sync"/>
						Subversion	file://home/andy/Documents/CV_WORK/SVN_REPOS/rep02				
<input type="checkbox"/>			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	cmbidge@no-reply	Subversion					
						Subversion					

To create a new bridge fill in the form above and click Apply.

If a Bridge needs to be disabled, locate the Bridge in question and click on the 'Change' checkbox. This will then enable the 'Active' checkbox and the ability to proceed with changing this value can continue. After changing the value, click the 'Apply' button and the Bridge will then be disabled. This means that even if the Subversion/Git/Mercurial repositories and/or ClearCase Views are changed with new files, modified files or deleted files, CM-Bridge will ignore the changes and continue with Bridges which are enabled. It is then possible to return at any time to the CM-Bridge web interface and re-enable the Bridge via the same method as disabling. After this has occurred, all changes which happened during the time when the Bridge was disabled will be synchronized until both sides of the Bridge are fully up-to-date.

Subversion Hooks

Prior to CM-Bridge version 2.0, Subversion *rev-prop* hooks were required to be installed on Subversion repositories. From Version 2.0 onwards, this is no longer the case.

ClearCase to Subversion/Git/Mercurial Bridges

Functionality

The ClearCase to Subversion, ClearCase to Git and ClearCase to Mercurial bridges provide the following functionality:

- Transfer of data in both directions
- Transfer of individual commits (between each bridge update) from Subversion to ClearCaseonly
 - This reflects the Subversion change history into ClearCase for commits that occurred between each bridge update
- Transfer of commit messages (comments) from ClearCase to Subversion/Git/Mercurial
- Transfer (where available) of committer, commit date and revision number from ClearCase to Subversion
 - These are transferred into the following Subversion Revision properties
 - `clearvision:cmbridge_remote_committer`
 - `clearvision:cmbridge_remote_timestamp`
 - `clearvision:cmbridge_remote_revision`

Considerations

A ClearCase to Subversion/Git/Mercurial Bridge system has four key elements:

1. ClearCase
2. Subversion and/or Git and/or Mercurial
3. CM-Bridge Server
4. CM-Bridge Webadmin Server

You need to consider the following points before setting up the CM-Bridge:

General

- The CM-Bridge server must be run on a supported platform.
- Both a ClearCase and a Subversion *client* must be installed on the CM-Bridge server host. `cleartool`, `git`, `hg` and `svn` must be in the `PATH` of the user id used for running the CM-Bridge server.
- On Linux systems, please set the user's `umask` to a suitable value so that ClearCase creates directories with suitable permissions. This is a common issue with ClearCase. If in doubt, please set the `umask` for the user id used for running the CM-Bridge server to '000'.
- The Subversion server can run on any machine accessible through the network from the CM-Bridge server host. There is no need to run the CM-Bridge server on the same host as the Subversion server.
- ClearCase VOBs can be hosted by any host within your network, provided they are accessible from the CM-Bridge server host.
- A special ClearCase attribute type must be created and applied to any VOBs being bridged
- There is no need for any CM-Bridge client software to be installed on ClearCase and Subversion client machines.
- All ClearCase, Subversion server and CM-Bridge server may reside on the same host. However, provided the points above are followed, you can set up the CM-Bridge as a standalone machine within your network.

ClearCase

Bridges connect ClearCase and Subversion/Git/Mercurial through a Subversion/Git/Mercurial workspace and a ClearCase view. Whilst the ClearCase view must be created manually, the CM-Bridge will automatically create internal Subversion/Git/Mercurial workspaces in the location specified by the `workspace` configuration option.

Note: ClearCase views need to have a suitable Config Spec. For Example, if you are bridging with the main line in ClearCase, the following config spec would be suitable:

```
element * CHECKEDOUT
element * /main/LATEST
```

⚠ Every ClearCase view used for a bridge must have `element * CHECKEDOUT` as the first rule to ensure correct operation.

⚠ Also, please use dedicated views for bridges, i.e. do **not** use another user's view for a bridge as otherwise the bridge could inadvertently

transfer checked out data or view-private files.

ClearCase Conflicts

In order to reduce conflicts on the ClearCase side, and in order to highlight the origin of the data, you could set up a special `subversion` branch to import Subversion data into. In addition to reducing synchronization conflicts, using a branch will also give you flexibility on how to use the data in views. What branch to use for the CM-Bridge is defined through the config spec of the view used for the bridge.

Subversion (for ClearCase to Subversion Bridges)

The CM-Bridge does not automatically start ClearCase VOBs and views or Subversion servers. Subversion must be up and running, ClearCase VOBs must be mounted and ClearCase views started before the CM-Bridge server is started.

The Subversion client software has been installed on the CM-Bridge Server host

The Subversion server should be up and running and access has been configured for the CM-Bridge user id

Git (for ClearCase to Git Bridges)

Git software must be installed on the CM-Bridge Server host.

The central (bare) Git repository is up and running; it has a `master` branch in it and access has been configured for the CM-Bridge user id.

Mercurial (for ClearCase to Mercurial Bridges)

Mercurial software must be installed on the CM-Bridge Server host.

The central Mercurial repository is up and running; it has a `default` branch in it and access has been configured for the CM-Bridge user id.

Multiple Bridges for the Same Repository

There are situations where it is useful to install multiple bridges between the same two databases. For example, you may choose to only bridge certain development branches of the ClearCase database with Subversion. This can be achieved by installing bridges on all branches to be bridged, configuring the ClearCase config specs for the workspaces to choose versions from specific branches.

When installing multiple bridges, it is strongly recommended to avoid overlap between the targets of different bridges. Take the following example:-

Bridge Name	Bridge Type	ClearCase View Path	Subversion/Git/Mercurial URL
Svn Bridge 1	Subversion	/view/bridge1_view/project_vob/src	http://svn.company.com/svn/project/trunk/src
Svn Bridge 2	Subversion	/view/bridge1_view/project_vob/src	http://svn.company.com/svn/project/branches/branch1/src

In the example above, the `src` data tree from the Subversion trunk as well as `branch1` from the branches area would both be bridged into the same view, i.e. the same branch in ClearCase. The result would be a mix of branch data and trunk data with no particular value.

What the user intended to do in the example is a valid use case. You can achieve the underlying goal of the example by creating two views with config specs for two different branches. For example you could bridge `.../trunk/src` into a view configured for the main line and then bridge `.../branches/branch1/src` into a view configured for a branch. This will keep main line (i.e. trunk) and branches in line between ClearCase and Subversion, i.e. data checked in to the ClearCase branch will be transferred into a Subversion branch, and the main line in ClearCase is transferred into the trunk in Subversion.

Bridge Name	Bridge Type	ClearCase View Path	Subversion/Git/Mercurial URL
Svn Bridge 1	Subversion	/view/bridge1_view/project_vob/src	http://svn.company.com/svn/proj/trunk/src
Svn Bridge 2	Subversion	/view/bridge2_view/project_vob/src	http://svn.company.com/svn/proj/branches/br1/src

The config spec for `bridge2_view` would be configured as follows in order for the view to operate on the branch:-

```
element * CHECKEDOUT
element * /main/branch1/LATEST
element * /main/LATEST -mkbranch branch1
```

Please bear in mind the following implications of creating a bridge that operates on a branch:-

- If the bridge is set up with no data in ClearCase and a branch in Subversion, the bridge will create branches for **every** element checked in to ClearCase. This is due to the branching model in Subversion effectively creating a copy of every element in the branch.
- If, however, the bridge is set up with no data in Subversion and a branch workspace in ClearCase, you are more likely to get the desired result, i.e. you only have branches in ClearCase for elements that have been changed on the branch.

On the Subversion side, you could use the CM-Bridge to only transfer particular baselines of a product using a bridge from the `tags` directory, or you could point to the trunk or particular branches in order to transfer development data. A combination of branches, tags and the trunk may also be useful. However, since the CM-Bridge does not translate between ClearCase and Subversion tagging concepts, you need to take care when setting up bridges for multiple tags.

Constraints and Limitations

The ClearCase to Subversion/Git/Mercurial Bridges have a number of constraints and limitations outlined below:

General

- The CM-Bridge does not translate between ClearCase and Subversion conventions. For example, Subversion uses `branches` and `tags` locations in the repository for management of branches and tags. When transferring data to ClearCase, these areas are treated as ordinary repository data with no special meaning attached.
- The CM-Bridge does not transfer checkin/commit log messages, branches, tags or other meta-data from SVN/Git/Mercurial to ClearCase. Only raw file data is transferred.
- The CM-Bridge requires ClearCase dynamic views and currently does not work with snapshot views.
- The account used for running the CM-Bridge server must have adequate access permissions to all areas to be bridged in ClearCase.


ClearCase

- ClearCase client software must be installed on the CMBridgeServer.
- The CM-Bridge does not automatically start ClearCase VOBs and views, so ClearCase VOBs must be mounted and the relevant ClearCase views started before the CM-Bridge server is started.
- On the ClearCase side any folder called `lost+found` at the top of a bridge will be ignored.
- The installation consists of three parts: the CM-Bridge server and Webadmin Server and ClearCase attribute type creation.

ClearCase Attribute

The CM-Bridge requires a special attribute type to exist for all ClearCase vobs being bridged. Please create the attribute type by changing your current working directory into the vob directory under a suitable view and run the following command:-

```
cleartool mkatype -vtype integer clearvision_cmbridge
```

 You will have to do this for ALL Vobs that are bridged, so please check that this has already been done *before* you add a new bridge!

Automatic creation of ClearCase Attribute

As of CM-Bridge v3.0, the ClearCase attribute will be created automatically within the view that is defined with your CM-Bridge Server config file.

Subversion (for CC2SVN bridges)

- The Subversion account specified in the bridge configuration must have adequate access permissions to all areas to be bridged in Subversion.
- SVN externals are not allowed within bridges.

Subversion to Subversion Bridges

Functionality

Subversion to Subversion Bridges provide the following functionality:

- Transfer of data in both directions
- Transfer of individual commits (between each bridge update) in both directions
 - This provides full history of the changes that occurred on each side of the bridge between bridge updates
- Transfer of commit messages
- Transfer of the following meta-data in both directions
 - Revision properties
 - File/folder properties
- Transfer of committer, time committed and remote revision in both directions
 - These are transferred into the following Subversion Revision properties
 - clearvision:cmbridge_remote_committer
 - clearvision:cmbridge_remote_timestamp
 - clearvision:cmbridge_remote_revision
- Control over what meta-data is transferred (using config file settings).

Considerations

A Subversion to Subversion Bridge system has three key elements:

1. Subversion
2. CM-Bridge Server
3. CM-Bridge Webadmin Server

You need to consider the following points before setting up the Bridge:

- The CM-Bridge server must be run on a supported platform.
- A Subversion *client* must be installed on the CM-Bridge server host and `svn` must be in the `PATH` of the user id used for running the CM-Bridge server.
- The Subversion server can run on any machine accessible through the network from the CM-Bridge server host. There is no need to run the CM-Bridge server on the same host as the Subversion server.
- There is no need for any CM-Bridge client software to be installed on Subversion client machines.
- The Subversion server and the CM-Bridge server may reside on the same host; However, provided the points above are followed, you can set up the CM-Bridge as a standalone machine within your network.

Multiple Bridges for the Same Repository

As with ClearCase to Subversion/Git/Mercurial, Subversion to Subversion also supports setting up multiple bridges for the same repository.

The scenario in this case could be one Subversion repository in one location sharing multiple projects with two different repositories located in different locations, e.g.

Bridge Name	Bridge Type	Subversion URL	Subversion URL
LondonToParis	Subversion	http://svn.london.com/svn/branches/paris/proj1/src	http://svn.paris.com/svn/proj1/trunk/src
NewYorkToParis	Subversion	http://svn.NewYork.com/svn/branches/paris/proj2/src	http://svn.paris.com/svn/proj2/trunk/src

Subversion Properties

Subversion to Subversion Bridges transfer File/Folder and revision properties between the repositories.

As it may not be relevant to transfer all properties between repositories, CM-Bridge provides the ability to ignore certain properties on transfer.

These are controlled through the newly introduced `Options` section of the configuration file.

```
[Options]
; This section outlines which properties should not be transferred between Subversion
; repositories on a Subversion to Subversion Bridge
svn_ignore_file_props=svn:mergeinfo,doc:language,
svn_ignore_rev_props=myprop,yourprop,hello:world
```

You need to consider the following points before adding/removing property ignore values:

- Each property is separated by a comma
- The tool (by default) will transfer every property not listed in the `...ignore...` lists, therefore certain properties should always remain in the list (see below)
- `svn_ignore_file_prop` applies to file **and** folder properties

Special Properties

Special consideration should be given to the impact of adding and removing properties from the ignore lists. For example `svn:merge-info` contains data specific to the repository for which it exists, therefore it should always remain in the list of ignored properties.

Another property example shown in the extract from the config file above shows a property called `doc:language`. In this case, the property could relate to the documentation language for a particular product and would therefore not be relevant in a repository being bridged to another country.

 `svn:externals` is not permitted in the bridge and is therefore not covered in this section

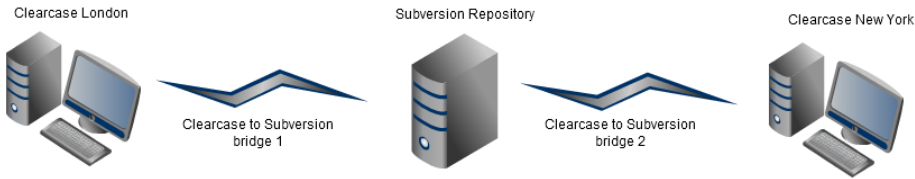
Constraints and Limitations

The Subversion to Subversion Bridge constraints and limitations are outlined below:

- SVN externals are not allowed within bridges.
- Subversion client software must be installed on the CM-Bridge Server host
- The Subversion account specified in the bridge configuration must have adequate access permissions to all areas to be bridged in Subversion.
- The CM-Bridge does not automatically start Subversion servers.
 - The Subversion server must be up and running (and access should be configured for the CM-Bridge user id) before the CM-Bridge server is started.

ClearCase to ClearCase bridges (Via Subversion)

CM-Bridge provides the functionality of bridging two ClearCase vobs via A Subversion repository. Note that this will always require two bridges to set up. In the diagram below a ClearCase vob at each end is bridged allowing the sharing of elements via a central subversion repository, In this scenario you must be prepared to deal with the subversion repository, As the bridge will always update ClearCase to subversion first any conflicts will always arise in subversion and not be passed onto the alternate vob until solved in the subversion repository. There are also extra considerations to take into account when using ClearCase UCM vobs, for further information on this please contact sales@clearvision-cm.com.



Multi-Site Bridges

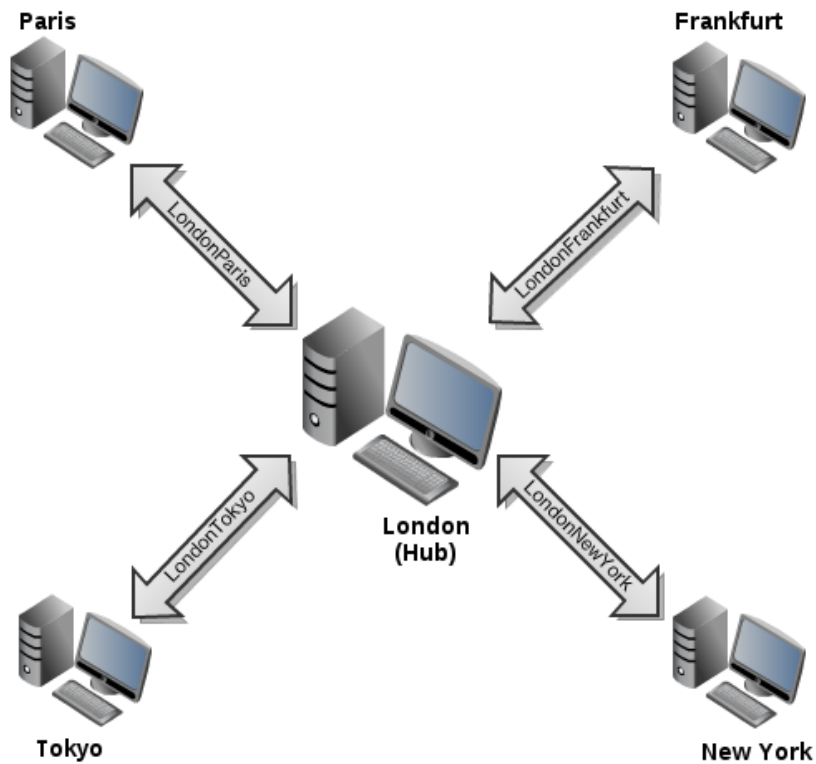
CM-Bridge v3.0 allows you to connect three or more Subversion-based sites from all over the World and keep them in sync (the same HEAD/LATEST revision). To use this functionality, read through the two examples below.

Star Setup

The Star design is the main supported Subversion multi-site setup as it requires the least number of full bridge update cycles to maintain synchronisation



This bridge set up will require 2 full bridge update cycles to ensure that the bridge is synchronised. This can be set in the CM-Bridge config file using the **update_cycle_factor** setting



Example Bridge Set up

The following table outlines each bridge that would typically be set up in the bridge admin website.


Bridge Name	Bridge Type	Bridge Left Side	Bridge Right Side
LondonFrankfurt	Subversion To Subversion	http://svn.london.com/svn/branches/frankfurt/proj1/src	https://svn.frankfurt.com/svn/proj1/trunk/src
LondonNewYork	Subversion To Subversion	svn://svn.london.com/svn/branches/newyork/proj1/src	http://svn.newyork.com/svn/projLondon/trunk/src
LondonTokyo	Subversion To Subversion	svn+ssh://svn.london.com/svn/branches/tokyo/proj1/src	svn+ssh://svn.london.com/svn/proj1/trunk/src
LondonParis	Subversion To Subversion	file:///home/svn/london/branches/paris/proj1/src	http://svn.paris.com/svn/proj1/trunk/src

Notes

1. With secure connections it may be necessary to accept ssh certificates permanently before setting up the bridge

Chain Design

A Chain Design can be used if you require ClearCase at one end of the bridge. In this scenario the ClearCase view **must** be at either end of the chain. It cannot be located within any bridge in the centre of the chain, otherwise changes passed into the ClearCase view will not be propagated any further along the bridge.

 In this set-up the number of bridge update cycles required to synchronise the entire bridge is equal to the number of links (in this case 4)



Example Bridge Set up

Bridge Name	Bridge Type	Bridge Left Side	Bridge Right Side	Note
LondonParis	ClearCase To Subversion	/views/londonBridge/vobs/london/project1/src	http://svn.paris.com/svn/trunk/project1/src	
ParisTokyo	Subversion To Subversion	http://svn.paris.com/svn/trunk/project1/src	http://svn.tokyo.com/svn/branches/project1/src	
TokyoNewYork	Subversion To Subversion	http://svn.tokyo.com/svn/branches/project1/src	https://svn.newyork.com/svn/trunk/project1/src	1
NewYorkFrankfurt	Subversion To Subversion	https://svn.newyork.com/svn/branches/project1/src	http://svn.frankfurt.com/svn/trunk/project1/src	1

Notes

1. This bridge shows a situation where only one side of the repository is https.

Merge Conflicts

Having setup Multi-Site Subversion Bridges, it is important to understand how CM-Bridge deals with Merge Conflicts within this environment. To demonstrate this, we are going to look at a hypothetical Merge Conflict between London and Paris.

- As it stands, all sites have synced correctly and each contain 3 files
 - file1
 - file2
 - file3
- John (in Paris), changes the contents of file1 from 'A message from Paris' to 'A message from John'
- Fred (in London), changes the contents of file1 from 'A message from Paris' to 'A message from Fred'
- They both commit their changes
- CM-Bridge performs two full update cycles

- A merge conflict is detected
- John sees the following content within file1

```
file1
<<<<<.mine
A message from Fred
----
A message from John
>>>>>.theirs (r blah blah)
```

- John confirms with Fred that he sees the same content in London

In this situation, it will be up to the Developers themselves to resolve the conflict and recommit the changes. CM-Bridge (v3.0+) will then perform another two pass update cycle and assuming the conflict has been resolved, all relevant changes will be sent across all Subversion sites.

Bridge Webadmin Configuration

Bridges are configured in the web-based graphical user interface. This section outlines how to start and use the CM-Bridge Webadmin site to set up your Bridges

CM-Bridge Webadmin Server

The Webadmin server must be started before the CM-Bridge server. This will create and configure important data structures required for the CM-Bridge server.

Please note that for the Webadmin server to operate correctly, you currently need to change your current working directory to the location of the Webadmin server executable and start the server from there (this will be fixed in a future release). You will also need to point to the location of the config file using the `-f` option:

```
cd /opt/Clearvision/cmbridge/admin
./cmbridge_webadmin -f /home/cmbridge/.cmbridge.cfg
```

After starting the Webadmin server, use a web browser to browse to the location of the admin server, e.g.: <http://my.server.com:8181>.

On Windows, you can use the Command Line to launch the `cmbridge_webadmin` executable (use the `-f` flag if the CM-bridge configuration file has been moved from the default location of the 'server' folder) or use the short-cuts created on the desktop (only present if you have used the CM-Bridge installer) and edit the 'target' parameter to add the `'-f C:\path\to\configfile.cfg'`.

Note: After you have added your bridges, you can safely stop the Webadmin server and leave the main server running (you only need the Webadmin Server running if you want to make changes).

You will then be presented with the following login page:-



The default user name and password for the CM-Bridge Webadmin server are `admin` and `admin`. Once logged in, you can change your password using the [Change Password](#) link at the top right of the screen.

If you typed the correct user name and password, you are presented with the Bridge Configuration view. In order to add a new bridge, fill in the form at the bottom of the page. The following table has some additional information on the fields to be filled in:

Field Name	Comments
Name	Please specify a unique name for the bridge.
Init Sync	Defines if you would like the server to perform an initial synchronization
Resolver Emails	A comma-separated list of email addresses to be notified if any problems occur.
Type	Specify the type of the bridge, i.e. Subversion, Mercurial or Git.
Repository (ClearCase/Svn)	Please enter the full absolute path/URL to the location within the ClearCase view/Subversion repository to be used for this bridge. Please note that the full path must exist on the file system.
Repository (Svn/Git/HG)	Please enter the full URL to the location within the Subversion/Git/Mercurial repository. Please note that the full path must exist on the file system.
User Name	Only applies to Subversion at this point. Please specify the user name to be used to log in to the Subversion repository. This can be left blank if you do not require authentication to log in to Subversion.
Password	Only applies to Subversion at this point. Please enter you Subversion password, or leave blank if no user authentication required.

ClearCase to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

Subversion to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input type="checkbox"/>				Subversion				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

ClearCase to Git Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Git				

To create a new bridge fill in the form above and click Apply.

In order to add new bridge, fill form at the bottom of the table and, when you are finished, press Apply. The newly configured bridge will then appear in the table as shown in the following screen-shot.

ClearCase to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input checked="" type="checkbox"/>	1	Golden gate	joseph.strauss@frisco.com	ClearCase	/views/san_francisco/vob/vob_name			<input type="checkbox"/>
<input type="checkbox"/>				Subversion	http://brooklyn.nyc.com/svn/	joseph	●●●●●●	
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

Subversion to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input type="checkbox"/>				Subversion				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

ClearCase to Git Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Git				

To create a new bridge fill in the form above and click Apply.

You can make certain changes to existing bridges. However, the only settings that you are currently allowed to change is the list of resolver email addresses and the Subversion user name and password. For Git/Mercurial, the only setting that can be changed is the resolver emails.

You update information for a bridge by ticking the box in the Change column of the table, changing the information directly in the table and clicking Apply at the bottom of the screen.

ClearCase to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input checked="" type="checkbox"/>	1	Golden gate	joseph.strauss@frisco.com	ClearCase	/views/san_francisco/vob/vob_name			<input type="checkbox"/>
<input type="checkbox"/>				Subversion	http://brooklyn.nyc.com/svn/	joseph	●●●●●●	
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

Subversion to Subversion Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input checked="" type="checkbox"/>	2	London	john.ennie@london.com	Subversion	http://city.london.com/svn	jrenni		<input type="checkbox"/>
<input type="checkbox"/>				Subversion	http://southwark.london.com/svn	rpmcculloch		
<input type="checkbox"/>				Subversion				
<input type="checkbox"/>				Subversion				

To create a new bridge fill in the form above and click Apply.

ClearCase to Git Bridges

Change	Id	Name	Resolver Emails	Side	Repository	User Name	Password	Delete
<input checked="" type="checkbox"/>	3	Brooklyn Bridge	emily.warren.roebing@nyc.com	ClearCase	/views/manhattan/vob/vob_name			<input type="checkbox"/>
<input type="checkbox"/>				Git	ssh://brooklyn.nyc.com/git			
<input type="checkbox"/>				ClearCase				
<input type="checkbox"/>				Git				

To create a new bridge fill in the form above and click Apply.

Note: within CC2GIT, CC2HG, CC2SVN and SVN2SVN bridges both ClearCase and Subversion sides can use whole repositories or a subdirectory within a repository. Note this is not possible within Git or Mercurial repositories which can instead be bridged to a separate branch. It is recommended that whole subversion repositories are NOT used as Subversion uses `branches` and `tags` locations in the repository for management of branches and tags. When transferring data to ClearCase, these areas are treated as ordinary repository data with no special meaning attached.

Should you need to delete a bridge, you can click on the icon to the right of the bridge. ⚠ Please note that deleting a bridge will not only stop the bridge but also reset internal meta-data. If you recreate a bridge for the same view and repository location at a later stage, the CM-Bridge server will re-transfer any data previously transferred. If you do need to delete and recreate a bridge, it is advised that you do not make any changes at either end of the bridge until the bridge has settled again (i.e. until the initial transfer has been completed).

Additional Users and Permissions

As of version 3.0+, the CM-Bridge Web Interface allows you to add additional users and specify if they have permission to perform a manual synchronisation on all or specific Bridges. To use this functionality, click on the 'Add User' link in the top right hand corner of the Web Interface (as seen below):

[Logout](#) [Change Password](#) [Add User](#) [Edit User](#)

After you have clicked this, the 'New User' page is shown (as seen below):



New User

User name:	<input type="text"/>
Password:	<input type="password"/>
Confirm Password:	<input type="password"/>

Control Sync Permissions:

Bridge	Allowed
london_to_paris	<input type="checkbox"/>

[Apply](#) [Cancel](#)

After filling in the required details, you then have the ability to pick and choose what Bridges the new User has access too. **Note:** They will be able to perform a manual synchronization at any time!

Manual Bridge Synchronization

Version 3.0 of CM-Bridge allows for users with the correct permissions to perform a Manual Synchronization of any specified Bridge. To carry out

this functionality, locate the Bridge you wish to update and click on the 'Sync' button. This will tell CM-Bridge to look at the differences between both sides of the bridge and transfer the changes accordingly.

Note: For Multi-site Bridges, this functionality will not synchronise the entire bridge 'network'. It will only synchronise the single bridge in question.

CM-Bridge Server

Before running the CM-Bridge Server, you should have already configured your Bridges using the Webadmin server and website.

Start the main server as follows:-

```
/opt/Clearvision/cmbridge/server/cmbridge_server -f /home/cmbridge/.cmbridge.cfg
```

The CM-Bridge server and Webadmin server **must** share the same config file to ensure that they are both configured to use the same workspace. It is via this workspace that the CM-Bridge server and Webadmin server exchange information.

Dealing with Merge Conflicts and Other Error Notifications

You are strongly advised not to ignore any error messages emailed to the list of resolver email addresses. If you receive such an email, it is very likely that the issues is permanent until resolved, even if you do not get any error messages again.

The most common issues are related to merge conflicts, which can occur if the same file is changed in Subversion/Git/Mercurial and ClearCase at the same time, and when the bridge comes to transfer changes across, it realises the conflict and informs the user.

If you get an email about a merge conflict, you can resolve the issue by checking in a fix changeset at either end of the bridge. The fixed files will then be transferred in the usual fashion with the next regular update and the two ends of the bridge will be back in sync.

Please refer to the User Guide for more information.

Troubleshooting

If you experience any issues during the installation please report your issues via the Clearvision Support Centre, which can be found at <http://www.clearvision-cm.com/support-home/support-center.html>.

This troubleshooting guide provides information on dealing with the most common issues that you may encounter.

Workspace Clean-up

Under rare circumstances it may become necessary to clean up the Subversion/Git/Mercurial working copies inside the CM-Bridge workspace. You can find the working copy for a bridge inside the workspace directory specified in the config file under `svn` (or `git/mercurial` respectively), within which you should see a subdirectory with the name of the bridge. Please do not delete the subdirectory but only inspect the working copy itself using `svn status` or `git status` or `hg status` and fix any problems you may see.

Forcing a Resynchronisation

If a large number of errors have occurred, and if you have already fixed the cause of the errors in the repository, it may be necessary to force a complete resynchronisation of a bridge. At the moment, this can only be achieved by deleting the bridge and recreating it with the same settings.

Please note that during a resynchronisation all data will be sent across again from both ends of the bridge, and it is therefore advisable to leave the bridge to settle before resuming work on the areas bridged. This is to prevent user changes from being overwritten by the bridge update.

Database Access Fails

If you see issues with locking of the `cmbridge.db` database file, then consider the following:-

- The `workspace` setting in the config file may have been configured to point to a network drive. There are known issues with network drives and file locking, which is required to work for database access. Please reconfigure this setting to point to a local file system.
- On Windows this may be caused by the `workspace` setting pointing to a drive with a FAT file system. If this is the case, please make sure that the windows share process `share.exe` is running in the task manager.

- On Linux, you may have started the CM-Bridge server and the Webadmin server under a different user id. If this is the case, please adjust file permissions on `cmbbridge.db` accordingly.

Appendix A: CC2Git on Windows

This section describes how to set up a remote Git repository hosted on Windows.

Prerequisites

There are two flavours of Git released for Windows Msysgit and Cygwin Git.

Msysgit is a native Microsoft Windows port for the Git client, which does not come with host services to allow the sharing of public repositories.

In order to use ssh or the git protocol to share a public repository you can use Cygwin. Although it is larger and slower, it does come with an ssh server and the git-daemon.

Note that you can use cygwin on the server and run git-daemon or ssh and still use msysgit on the clients.

Steps

This example will use the ssh protocol.

Server or remote repository

Setup

- Install Cygwin
 - Select Packages
 - Devel > git
 - Devel > git-completion
 - Devel > git-gui
 - Devel > gitk
 - Net > openssh
 - Python > python
- run Cygwin
- Configure openSSH
 - `run ssh-host-config`
 - Follow onscreen instructions
- Start SSH
 - `net start sshd`
- Open Windows Firewall and create an exception to allow TCP traffic on port 22.

Create Repo

When pushing to a remote git repository, it is easier if you push to a bare repository `$ git --bare init }}`. If you want to push to a non-bare repository, you need to do a `{{ $ git reset --hard HEAD and $ git checkout -f on the remote server to sync the heads up. The latter is only recommended if you need to interact directly with the files on the remote server.`

The issue seems to be designed to prevent changes made locally on the remote repository being lost when changes are pushed from elsewhere.

```
git --bare init
```

Client or local repository

Setup

- Install Cygwin or Msysgit
- Setup public keys
- Navigate to directory you wish to store repository
- Clone Git repository `git clone ssh://hostname.domain/path/to/repo.`

If you receive "warning: remote HEAD refers to nonexistent ref, unable to checkout." the first time you clone a remote repository, this is likely to be because you are cloning a bare repository. Adding, committing and pushing a file should resolve any problems.

Appendix B: Setting up SSH Keys for automatic login to Git SSH repositories

In order for CC2GIT to work with ssh Git repositories it is necessary to set up ssh Keys so that the client machine can perform an automatic login to the repository.

This is necessary because CC2GIT works as a server process and it will not respond to a login request from the ssh server when accessing the Git Repository.

Prerequisites

Linux

Ensure that both Git and ssh client tools are installed as per your distribution.

Windows

Ensure that either msysgit or Cygwin Git are installed (See Appendix A)

Setup

Linux

- Open a terminal shell
- Run the following command
 - `ssh-keygen -t rsa`
 - Accept the default store for the keys
- Press `enter` when prompted to create a password. This will create an empty password.
- Run the following command to copy your key to the remote machine
 - `ssh-copy-id <username>@<remote machine>`
 - e.g. `ssh-copy-id paulsmith@mysshrepo.com`

Windows

The msysgit and cygwin Git installations work differently when setting up the ssh keys. Please follow the relevant instructions based on your client install of Git

msysgit

- Open the Git Bash Shell
 - Start -> All Programs -> Git -> Git Bash
- Run the following command
 - `ssh-keygen.exe -t rsa`
 - Accept the default store for the keys
- Press `enter` when prompted to create a password. This will create an empty password.
- This will place two files in the `.ssh` directory `id_rsa` and `id_rsa.pub`
- Copy the contents of the `id_rsa.pub` file and append them to the end of the `authorized_keysfile` on the remote ssh server
 - Note: You may need to get the remote ssh server's administrator to do this for you.

Cygwin Git

Cygwin

- Open the cygwin shell
 - Start -> All Programs -> Cygwin -> Cygwin Bash Shell
 - Run the following command
 - `ssh-keygen.exe -t rsa`
 - Accept the default store for the keys
- Press `enter` when prompted to create a password. This will create an empty password.
- Run the following command to copy your key to the remote machine
 - `ssh-copy-id <username>@<remote machine>`
 - e.g. `ssh-copy-id paulsmith@mysshrepo.com`