SERENA®

DIMENSIONS® CM 14.3.3

Git Connector User's Guide

Serena Proprietary and Confidential Information

Publication date: May 2017

Table of Contents

Chapter 1

Overview

About the Dimensions CM Git Connector

The Dimensions CM Git Connector brings central control and security to teams using Git, allowing them to store code in a CM stream. Streams can be shared by developers using Git or Dimensions CM. The connector is a command-line tool.

- Use the `clone` command to clone a stream from Dimensions CM and populate the master branch in a Git repository with the tip of a stream.
- Use the `fetch` and `pull` commands to update a local Git repository with changes from Dimensions CM. When you pull changes, each CM changeset becomes a commit in a local Git repository.
- Use the `push` command to push changes from a Git repository into Dimensions CM. Each Git commit becomes a Dimensions CM changeset, including the user ID and timestamp.

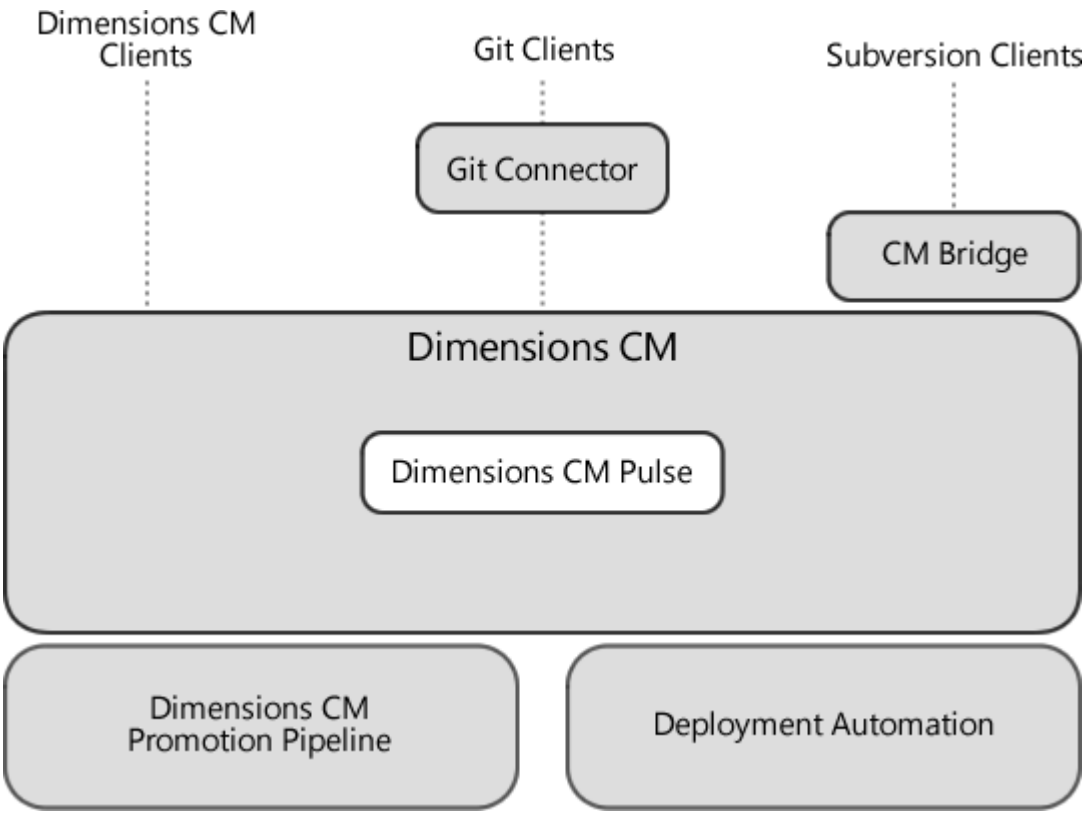
NOTE

- Git users are automatically registered as CM users when their commits are pushed.
- Empty folders in Dimensions CM are not added to a Git repository.

Architecture

Dimensions CM is an enterprise SCCM (Software Configuration and Change Management) repository that securely stores code and artifacts from teams using Git, Subversion, and Dimensions CM. Additional features include:

- Peer review and continuous inspection with Dimensions CM Pulse.
- Automation of the path to production using the Promotion Pipeline and Deployment Automation.

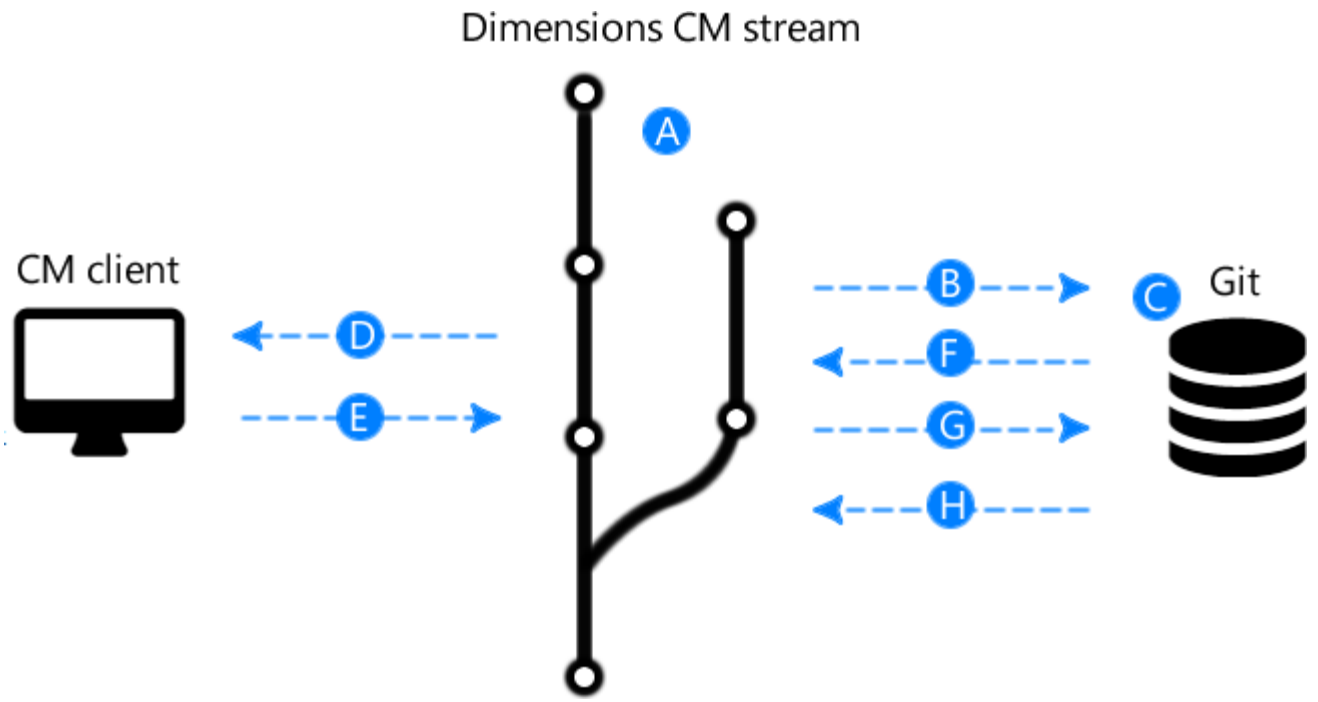


Installing and Licensing

The Dimensions CM Git Connector is a separate installer that you can download from the Serena Support web site.

IMPORTANT! The connector requires an additional license.

Typical Workflow



A A development stream is under the central control of Dimensions CM.

B A GIT developer clones the stream into their local GIT repository.

C The GIT developer branches, commits, and works as normal with their GIT repository.

D Another developer is using a Dimensions CM client. They update their local CM controlled work area from the same stream.

E The second developer works on the code and delivers their changes to Dimensions CM.

F The GIT developer tries to deliver their changes. The deliver fails as there are changes in the stream that they need to merge.

G The GIT developer pulls the changes and merges them into their local GIT repository.

H The Git developer can now successfully deliver their changes to Dimensions CM.

Request Management

You can specify one or more Dimensions CM requests when you commit changes to your local Git repository. Specify the requests IDs in the following format in the commit comment (separate request IDs with a comma):

```
git commit -m "[<request ID>,<request ID>] <your commit message>"
```

When you push changes from a Git repository to Dimensions CM you can override the requests you specified during the commit by using the `--requestids` parameter. For detail see page 16.

Using CAC with the Git Connector

If you are going to use Common Access Card (CAC) with the Git Connector:

1 Open the following file:

`%USERPROFILE%\.gitdm\gitdm.properties`

For instance, if you are logged in as jsmith, open:

`C:\Users\jsmith\.gitdm\gitdm.properties`

Locate these lines:

```
# Smart Card library path

activclient-location=
```

Change the line to point to the location of your ActivClient PKCS DLL file, for example:

```
# Smart Card library path

activclient-location=C:\\Program
Files\\ActivIdentity\\ActivClient\\acpkcs211.dll
```

2 Add the server's public certificate, and any certificates in the chain, to the correct `cacerts` file.

To confirm which keystore you need to update, open `%DM_ROOT%\prog\git-dm.cmd` and check which JAVA_HOME the Git Connector is using. Verify that this JAVA_HOME `cacerts` file has been updated correctly.

Chapter 2

Command Reference

Command Syntax

```
git-dm <command> <parameters>
```

General Parameters

The following optional parameters are available for each command:

`--card`

Use smart card authentication.

`--help`

Displays help for the current command.

`--password`

Enter a password for a CM user. Not required if you are using a smart card.

`--quiet`

Does not display the progress of the output.

`--username`

Enter a CM user name. Not required if you are using a smart card.

`--verbose`

Displays the progress of the output.

`--version`

Displays the version of the Git Connector.

Reference

clone

```
git-dm clone <server> <product:stream> [directory]
```

Description

Initializes a Git repository from a Dimensions CM stream.

Parameters and Qualifiers

`<server>`

Specifies the Dimensions CM server and database connection in the following format:

`scm:dimensions://hostname/dbname@dbconnection`

`<product:stream>`

Specifies the Dimensions CM product and stream in the following format:

`product:stream`

`[directory]`

(Optional) Specifies a directory where the stream will be cloned.

Example

```
git-dm clone
scm:dimensions://cmserver/cm_typical@dim14 QLARIUS:SAVINGS
C:\temp\savings --username dinesh -password <password>
```

This example clones the stream `SAVINGS` in the product `QLARIUS` from the Dimensions CM server `cmserver` and the base database `cm_typical@DIM14`. The contents of the stream are cloned into a new Git repository in the folder `C:\temp\savings`. The command is performed as the user `dinesh`.

fetch

```
git-dm fetch
```

Description

Fetches the latest code from a Dimensions CM stream into the FETCH_HEAD in the local Git repository.

Parameters and Qualifiers

`--deep`

(Default) Creates commits for each Dimensions CM changeset since the last fetch you performed.

`--force`

Forces a fetch of the last Dimensions CM changeset that you downloaded.

`--shallow`

Creates a single commit for all the CM changesets.

Example

```
git-dm fetch
scm:dimensions://cmserver/cm_typical@dim14 QLARIUS:SAVINGS
--force --card
```

This example fetches any changes made in Dimensions CM into the local Git repository and specifies multiple CM requests. The `--force` option overwrites any local changes that have not been pushed to Dimensions CM. The `--card` option prompts the user to insert their CAC card, select a certificate, and enter their PIN to log into Dimensions CM.

pull

```
git-dm pull
```

Description

Pulls the latest code from a Dimensions CM stream and merges the changes into the Git master branch.

Parameters and Qualifiers

`--deep`

Creates commits for each Dimensions CM changeset since the last fetch you performed.

`--force`

Forces a fetch of the last Dimensions CM changeset that you downloaded.

`--ours`

Preserves local changes when merging.

`--rebase`

Rebases a Git master branch on top of the latest changes from Dimensions CM after a fetch.

`--resolve`

(Default) Resolve conflicts when merging.

`--shallow`

Creates a single commit for all the CM changesets.

`--theirs`

Preserves remote changes when merging.

Example

```
git-dm pull
scm:dimensions://cmserver/cm_typical@dim14 QLARIUS:SAVINGS
--shallow --ours --username dinesh -password <password>
```

This example pulls any changes made in Dimensions CM into the local Git repository. The `--shallow` option creates a single commit in the local Git repository for all the changesets that were created in CM. If conflicts are found during the pull, the `--ours` option ensures that the local changes are preserved and not merged or overwritten by the remote changes. The command is performed as the user `dinesh`.

push

```
git-dm push
```

Description

Pushes changes to a Dimensions CM stream from a Git repository.

NOTE Only the tip changeset generated by a `git-dm push` operation has a peer review and executes expert chains.

Parameters and Qualifiers

`--autoignore`

Automatically selects the commit paths in the Git repository tree to be ignored when a commit has more than one parent resulting from a merge.

`--deep`

Creates commits for each Dimensions CM changeset since the last fetch you performed.

`--message <text>`

Specifies a message to be added as a comment to the CM changeset.

`--metadata`

Includes the Git commit metadata in the changeset (use with `--deep`).

`--no-metadata`

(Default) Does not include the Git commit metadata in the changeset (use with `--deep`).

`--renamemode <RenameMode>`

Specifies a rename mode to use when pending changes. Specify one of the following:

`all`

`justFiles` (default)

`none`

`--requestids <CM request ID>`

When you push changes from a Git repository into Dimensions CM, use this parameter to override the requests previously specified for Git. If you use `--deep` the requests specified as a command line option are added to the requests specified in the commit comment (if any).

`--shallow`

Creates a single commit for all the CM changesets.

`--ignore`

Specifies commit IDs to be ignored when performing a deep push if one commit has more than one parent resulting from a merge. For example:

```
--ignore=id1,id2
```

Example

```
git-dm push --requestids QLARIUS_CR_39 --username dinesh
    -password <password>
```

This example pushes changes from the local Git repository into Dimensions CM. Any changesets that are created are related to the CM request QLARIUS_CR_39. The command is performed as the user `dinesh`.

Privileges

The user performing the push operation must have the following privilege:

Deliver files into project/stream

If a user makes a commit to Git but is not a registered CM user, then the user performing the push of the commit must have the following privilege. This is required for the user to be automatically registered with CM:

Manage Users and Group Definitions

For more details see the *Process Configuration Guide*.