MICRO FOCUS®

Dimensions CM

Git Client User's Guide

Product version: 2.1

Last updated: August 20, 2019

Table of Contents

Chapter 1

Overview

About the Dimensions CM Git Client

The Dimensions CM Git Client brings central control and security to teams using Git, allowing them to store code in a CM stream while enjoying the familiar Git user experience. Streams can be shared by developers using Git or Dimensions CM.

The Dimensions CM Git Client is an extension to Git allowing the use of the standard Git command line, Integrated Development Environments (IDEs), and other tools using Dimensions CM as the remote repository. For example, you can:

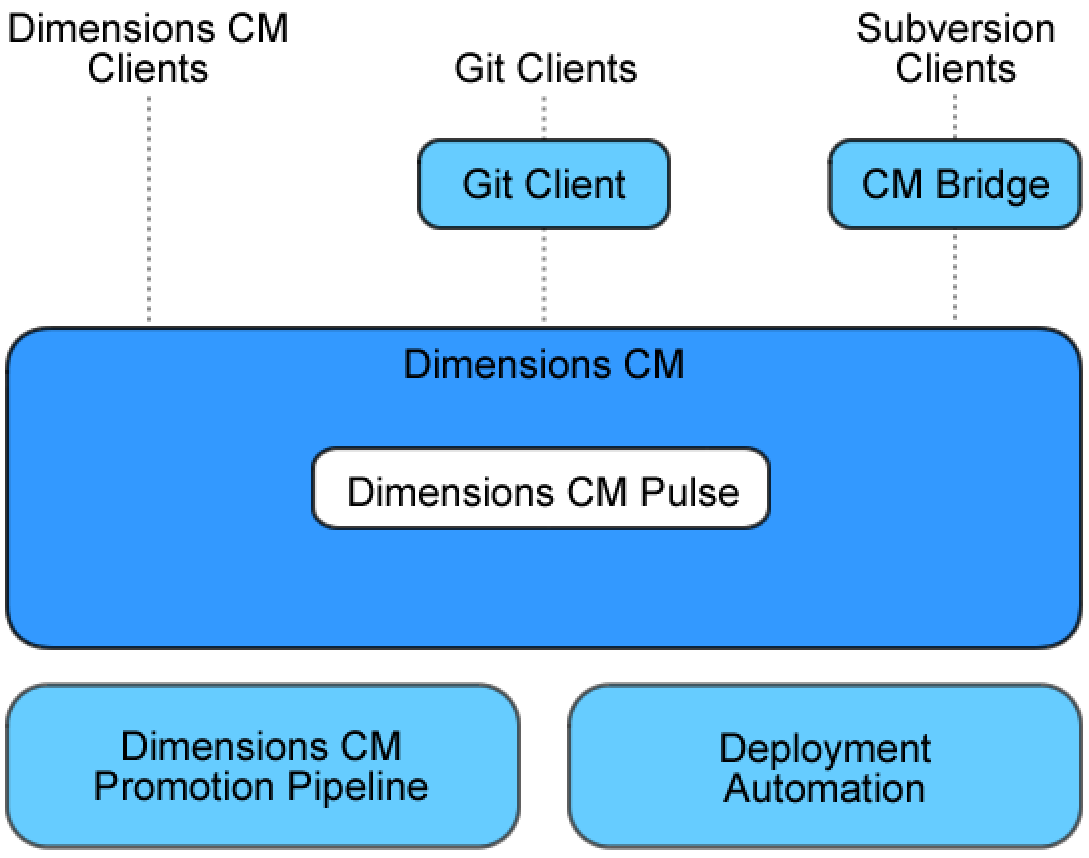
- Use the `clone` command to clone streams from Dimensions CM and populate a local Git repository.
- Use the `fetch` and `pull` commands to update a local Git repository with changes from Dimensions CM. When you pull changes, each CM changeset becomes a Git commit.
- Use Git to branch, commit, and merge.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

Git users are automatically registered as Dimensions CM users when their commits are pushed.

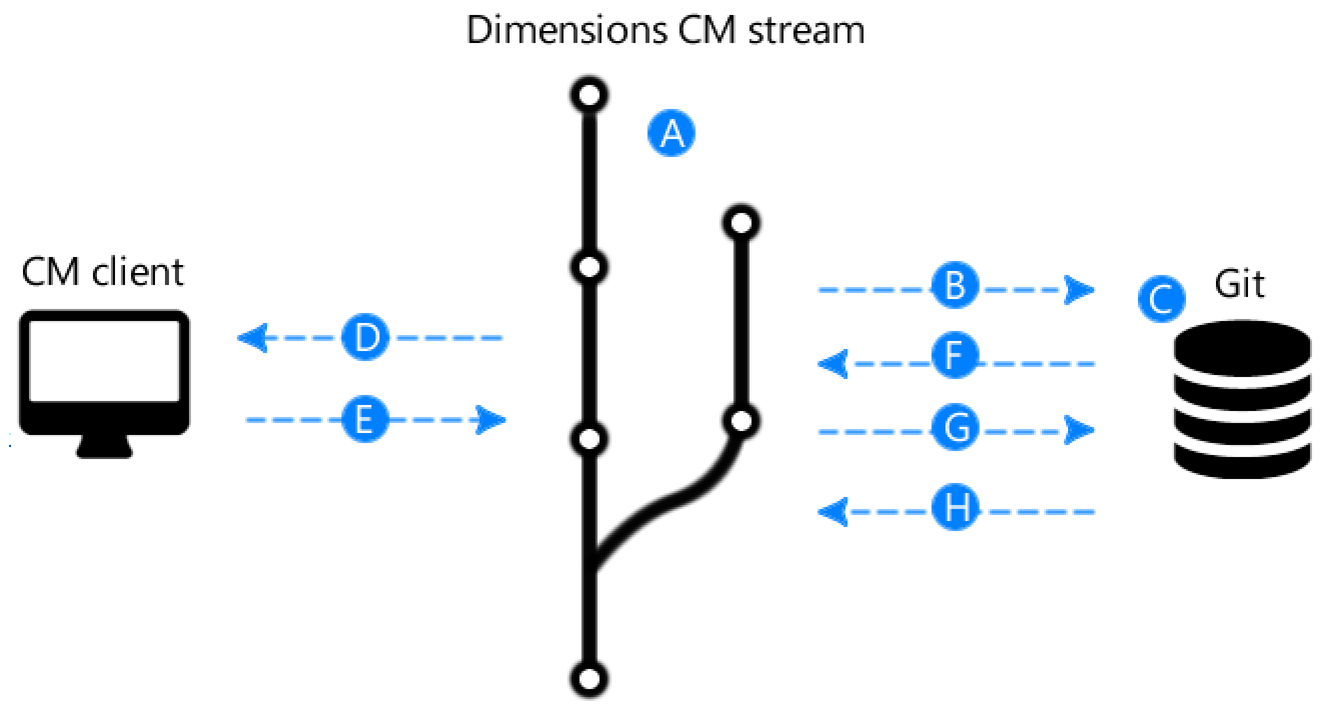
Architecture

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 Micro Focus Pulse.
- Automation of the path to production using the Promotion Pipeline and Deployment Automation.



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.

Chapter 2
Using the Git Client

Storing Credentials

The Dimensions CM Git Client requires user credentials to perform some operations, for example, to log into a CM server. Optionally, use one of the following methods to securely store your credentials so that you do not have to re-enter them:

- Short term: cache credentials in memory.
- Long term: store credentials using a configurable third-party application that integrates with the secure storage provided by your operating system.

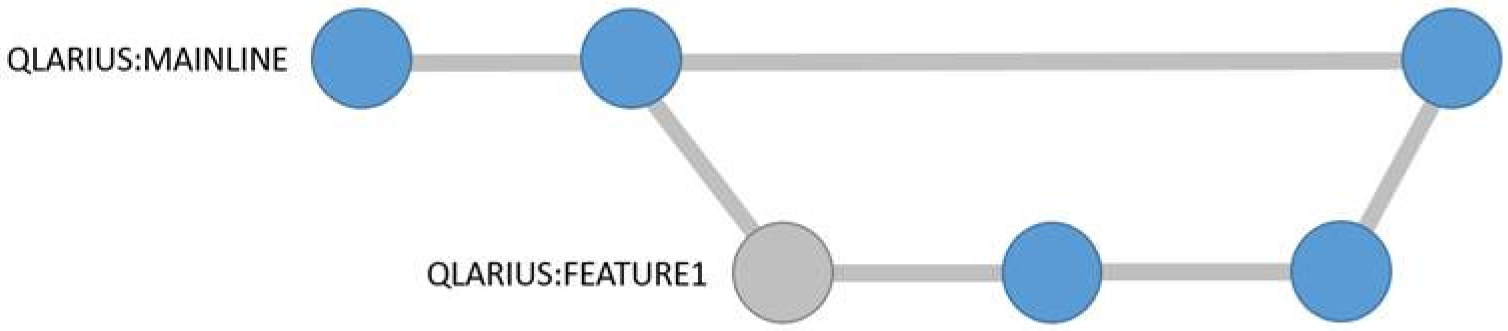
For details see this Git web page: https://git-scm.com/docs/gitcredentials

Branch and Merging

The Git Client allows you to branch and merge using standard Git functionality. Git supports many different branching models and types of merge.

Branching

When a stream is cloned from Dimensions CM it becomes a branch in your local git repository. New local branches pushed back into Dimensions CM become new streams. For example, assume you have the stream `QLARIUS:MAINLINE` with another stream branched from it, `QLARIUS:FEATURE1`.



You can clone these streams into your local git repository using this command:

```
git clone dimensions://servername/cm_typical@dim14/qlarius/mainline/* .
```

The master branch in Git contains the files from the stream `QLARIUS:MAINLINE` stream. There is also a branch called `feature1` containing the files from `QLARIUS:FEATURE1`. You can work on these branches in Git just like any other Git repository. For example, switch to the `feature1` branch:

```
git checkout feature1
```

Make code changes, such as edit files and add new files, and commit them locally:

```
git commit -m “made some local changes”
```

Push that commit to Dimensions CM into the `QLARIUS:FEATURE1` stream:

```
git push
```

Creating New Streams

You can create branches locally and push them as new streams to Dimensions CM. Continuing from the example above, assume that you want to create a new branch from the mainline called `feature2`. First, switch to the master branch:

```
git checkout master
```

Next, create the new local branch:

```
git checkout -b feature2
```

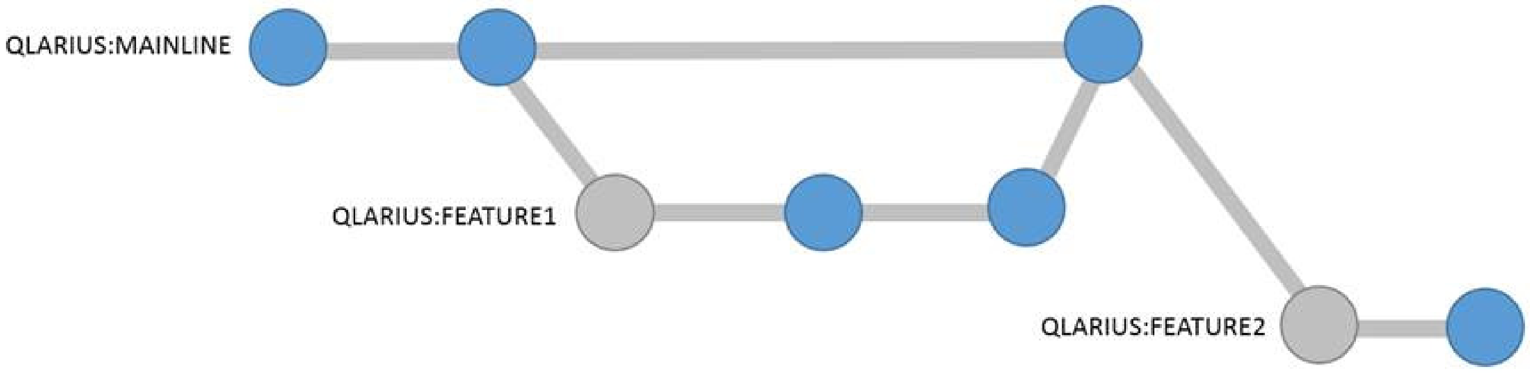
Make changes in the new local branch and commit them:

```
git commit -m “made some changes in a new branch”
```

Push the new branch to Dimensions CM, which creates a new stream. You must push and set the upstream reference to the name of the new stream:

```
git push --set-upstream origin feature2
```

This creates a new stream `QLARIUS:FEATURE2` and delivers the changes to it. The streams in Dimensions CM now look like this:



NOTE Streams created by the Dimensions CM Git Client by default are topic streams, which use pull requests for reviewing and merging changes. For details about topic streams see the *Dimensions CM User’s Guide*. You can also create normal streams, for details see page 19.

Merging

You can merge branches in your local Git repository and push the results of the merge back to Dimensions CM. Continuing from the example above, make changes on the master branch and `feature2` branch, merge, and push. First, switch to the `feature2` branch:

```
git checkout feature2
```

Make changes and commit them:

```
git commit -m “made feature2 changes”
```

Switch to the master branch:

```
git checkout master
```

Make changes and commit them:

```
git commit -m “made mainline changes”
```

Merge `feature2` into the mainline using Git:

```
git merge feature2
```

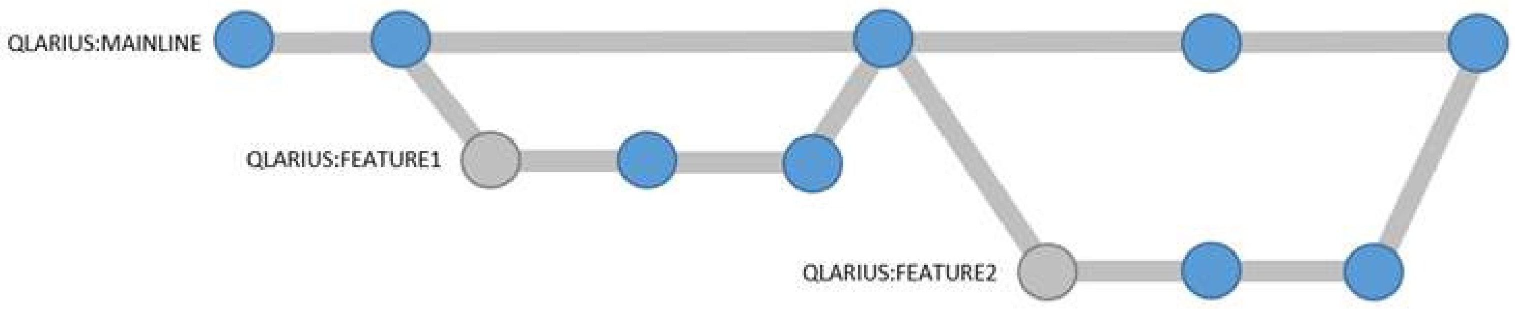
Resolve any merge conflicts and commit:

```
git commit –m “merged feature2 into mainline”
```

Push all the branches back to Dimensions CM:

```
git push --all origin
```

The new changes are delivered to both streams and the streams are merged. The streams in Dimensions CM now look this:



NOTE If you perform the merge in Dimensions CM it will be recorded in Git next time you pull or clone the streams.

Managing Requests

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>"
```

Git Client Command Line Format

The Dimensions CM Git Client uses the standard Git command line format:

```
git <command> <URL>
```

where:

`<command>`

Specifies a Git command, see the online reference: https://git-scm.com/docs

`<URL>`

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

```
dimensions://username:password@hostname:[port number]
/databasename@connection/product/stream/[directory]
```

where:

`[port number]` specifies an optional port number on the server.

`[directory]` specifies an optional path to a directory.

Example URL:

```
dimensions://dinesh:mypassword@cmserver:8080/cm_typical@dim14/
qlarius/savings
```

If you are cloning a specific stream version on Linux or using Git Bash, add quotes to the URL, for example:

```
git clone ‘dimensions://dmsys:dmsys_test@myserver/cm_typical@dim12
/PROD/ST1;0’
```

Authenticating using Common Access Cards

You can use a Common Access Card (CAC) to authenticate a connection to a CM server. Use `dimensionscac` in the URL instead of `dimensions`:

```
dimensionscac://<servername>[:<port>]/<base database>@<connection>/
<product>/<stream>[/<folder>]
```

Example Workflow

1 Insert a CAC card into the reader.

2 Run a `clone` command, for example:

```
git clone dimensionscac://myserver/cm_typical@dim14/qlarius/
java_brancha_str .
```

3 If promoted, enter the card's PIN.

4 If there are multiple certificates on the card, select one.

5 The content is now cloned.

Changing the Location of the ActivClient Library

The installer automatically detects your ActivClient installation and configures Git Client to use it. To manually configure the location of the ActivClient PKCS#11 library:

1 Edit this file:

- Windows

  ```
  %GIT_HOME%\mingw64\libexec\git-core\start-git-client
  ```

- UNIX, Linux, and MacOS

  ```
  /usr/libexec/git-core/start-git-client
  ```

2 Locate this line:

```
export ACTIV_PATH=
```

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

```
export ACTIV_PATH=C:\Program
Files\ActivIdentity\ActivClient\acpkcs211.dll
```

Supported ActivClient Versions

- ActivClient 7.0.2 hotfix 409
- ActivClient 7.1

Command Reference

clone

```
git clone <URL> <directory>
```

Description

Creates a new Git repository from one or more Dimensions CM streams.

Examples

```
git clone dimensions://cmserver/cm_typical@dim14/qlarius/savings
C:\temp\savings
```

Clones the stream `SAVINGS` in the product `QLARIUS` from the Dimensions CM server `cmserver` and the base database `cm_typical@DIM14`. The content of the stream is cloned into a new Git repository in the folder `C:\temp\savings`. The command should prompt for credentials when first used.

```
git clone dimensions://dinesh:<password>@cmserver/cm_typical@dim14/
qlarius/savings/Documents C:\temp\savings
```

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

```
git clone dimensions://cmserver/cm_typical@dim14/qlarius/
mainline_java_str/* C:\temp\work
```

Clones the stream `MAINLINE_JAVA_STR` and all of its child streams in the product `QLARIUS` using the Dimensions CM server `cmserver` and the base database `cm_typical@DIM14`. The contents of the stream `MAINLINE_JAVA_STR` are cloned into a new Git repository in the local folder `C:\temp\work`. Each child stream is cloned into a separate branch in the Git repository.

Cloning Multiple Streams

If you have multiple streams you can clone them in one operation. For example, your team has a collection of streams that they are frequently cloning.

1. Create a .git text file, for example: `myStreams.git`
2. Add each stream to the file on a separate line in the format `<PRODUCT>:<STREAM NAME>`, for example:

   ```
   QLARIUS:PARENT_STREAM
   QLARIUS:CHILD_STREAM
   ```

   The first stream in the list becomes the master branch in Git.
3. Deliver the .git text file into a Dimensions CM stream.
4. Use the .git text filename in the Git command, for example:

   ```
   git clone dimensions://myserver/mydatabase@dbconn/qlarius/mystream/
   myStreams.git
   ```

TIP

- To specify multiple streams use wildcards, for example:

  ```
  QLARIUS:OTHER*
  QLARIUS:*JAVA*
  ```
- If your team is using the same collection of steams, they can share the .git text file.

fetch

```
git fetch
```

Description

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

Example

```
git fetch --all
```

Fetches any changes made in Dimensions CM into the corresponding branches of the local Git repository. The `--all` option performs the fetch in all branches that correspond to Dimensions CM Streams rather than just the current branch. Each new changeset that is fetched becomes a commit in the local Git repository.

pull

```
git pull
```

Description

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

Example

```
git pull --strategy=ours
```

Pulls any changes made in Dimensions CM and merges them into the local Git repository. If conflicts are found during the pull, the `--strategy=ours` option ensures that the local changes are preserved and not merged or overwritten by the remote changes.

push

```
git push
```

Description

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

NOTE

- Only the tip changeset generated by a `git push` operation has a peer review and executes expert chains.
- By default, streams created by the Git Client are topic streams. Use the option `topic=false` to create a normal Dimensions CM stream.
- By default, requests are not related to topic streams. Use the option `topicrequest=<request ID>` to relate a request.

Examples

```
git push
```

Pushes changes from the current branch of the local Git repository into Dimensions CM. Each commit that is pushed becomes a Dimensions CM changeset in the corresponding stream.

```
git push --set-upstream origin newstream
```

Creates a new topic stream called `newstream` and pushes changes from the current branch of the local Git repository into that stream.

```
git push --set-upstream origin newstream --push-option="topic=false"
```

Creates a new normal stream called ’newstream’.

```
git push --set-upstream origin newstream
--push-option="topicrequest=QLARIUS_CR_36"
```

Creates a new topic stream called ’newstream’ and relates it to request ’QLARIUS_CR_36’.