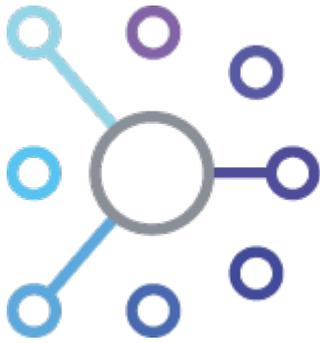


DLCM Tools Guide

Table of Contents

1. Tool Options	2
1.1. <code>load</code> function	2
1.2. <code>local-load</code> function	2
1.3. <code>check</code> function	2
1.4. <code>import</code> function	2
1.5. <code>git</code> function	2
2. Installation	3
2.1. Get the tool	3
2.2. Organize the data	4
3. Execution	6

DLCM Tools v2.2.7, 2023-12-04



DLCM

The current documentation is available in [HTML](#) or [PDF](#).

Chapter 1. Tool Options

1.1. load function

This function allows to add data files to an existing deposit, identified by `<Deposit ResId>`.

For each deposit, the tool will crawl the corresponding folder of the deposit and upload each file as a data file in the correct folder (i.e. `relative location`).



The upload volume is limited. Generally, the limitation is up to 1GB: : ask to administrator of the service.

1.2. local-load function

This function is equivalent the `load` one by referencing files instead of uploading them.

It is recommended for big files, greater than 1GB.



The prerequisite is to transfer files on `pre-ingest` machine, before to run the tool.

1.3. check function

This function allows to verify if there is the same number of files by comparing the folder on disk and the deposit.

1.4. import function

This function allows:

1. To create an organizational unit, if it does not exist. The control criterion is the name of organizational unit
2. To create a deposit with the folder name as title
3. To add data files by uploading files with relative location

1.5. git function

This function allows to create a deposit and to add data files from a GIT repository.

The extraction is based on a branch or a tag or a commit.

Chapter 2. Installation

2.1. Get the tool

- Download the DLCM tool package: [DLCM-Tools-2.2.7.jar](#)
- Define the properties file

dlcm-tools.properties

```
#####
# 'load' 'local-load' & 'check' parameter # ①
# Folder where the data to load or check are #
#####
dlcm.load=<Must be defined>

#####
# 'import' parameter # ②
# Folder where the data to import are #
#####
dlcm.import=<Must be defined>

#####
# 'git' parameters # ③
#####
dlcm.git.organizational-unit=<To be defined>
# Example: dlcm.git.organizational-unit=DLCM
dlcm.git.repository-url=<Must be defined>
# Example: dlcm.git.repository-url=git@gitlab.unige.ch:solidify/solidify-tools.git
dlcm.git.branch=<Could be defined, optional>
# Example: dlcm.git.branch=1.3-maintenance
dlcm.git.commit=<Could be defined, optional>
# Example: dlcm.git.commit=729a871d
dlcm.git.tag=<Could be defined, optional>
# Example: dlcm.git.tag=dlcm-1.3.1
#-----#
# Could be changed #
#-----#
dlcm.git.data-category=Software
dlcm.git.data-sub-category=Code

#####
# 'purge' parameters # ④
# List of relative locations to purge #
#####
dlcm.load=<Must be defined>
dlcm.relative-locations=<Must be defined>
# Example to purge all data files: dlcm.relative-locations=/
# Example to purge data files in images & old: dlcm.relative-locations=/images,/old
```

```
#####
# Profile to determine which function to run #
# - load = To upload data files in existing deposits #
# - check = To check between file system & existing deposits #
# - import = To create deposit and to upload data files #
# - git = To import data files from GIT repository #
#####
spring.profiles.active=<Must be defined> ⑤
# Examples:
# spring.profiles.active=load
# spring.profiles.active=import,git

# DLCM modules
dlcm.module.admin.url=<DLCM Admin module URL: must be defined>
# Example: dlcm.module.admin.url=https://sandbox.dlcm.ch/administration/admin
dlcm.module.preingest.url=<DLCM Preingest module URL: must be defined>
# Example: dlcm.module.preingest.url=https://sandbox.dlcm.ch/ingestion/preingest
dlcm.module.ingest.url=
# Example: dlcm.module.ingest.url=https://sandbox.dlcm.ch/ingestion/ingest
dlcm.module.archival-storage.urls=
# Example: dlcm.module.archival-
storage.urls=https://sandbox.dlcm.chstoragion/archival-storage

spring.cloud.config.enabled=false

# Client should not instantiate JDBC / Hibernate / JPA layer
spring.autoconfigure.exclude[0]=org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAutoConfiguration
spring.autoconfigure.exclude[1]=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration
spring.autoconfigure.exclude[2]=org.springframework.boot.autoconfigure.jdbc.DataSourceTransactionManagerAutoConfiguration
spring.autoconfigure.exclude[3]=org.springframework.boot.autoconfigure.data.web.SpringDataWebAutoConfiguration
```

- ① Parameters to upload/reference data files in existing deposits **or** to check between file system & existing deposits
- ② Parameters to create deposit and to upload data files
- ③ Parameters to import data files from GIT repository
- ④ Parameters to purge all data files in relative locations
- ⑤ Profiles to run tools: **load** / **local-load** / **check** / **import** / **git** / **purge**

2.2. Organize the data

- The structure of the **import** folder must be like this:

<Organisational Unit Name> / <Deposit Tile> / <Data Category> / <Data Type> / Your Data

or

<Organisational Unit Name> / <Deposit Title> / Package / CustomMetadata / <Metadata Type resId>
/ Your Data

- The structure of the `load` / `local-load` / `check` folder must be like this:

<Deposit ResId> / <Data Category> / <Data Type> / Your Data

or

<Deposit ResId> / Package / CustomMetadata / <Metadata Type resId> / Your Data



Data Category & Data Type values: see [Data Categories & Types](#)

Chapter 3. Execution

To run the tool:

- Set the environment variable to define OAuth2 access token

```
export OAUTH2_TOKEN=<OAuth2 Access Token>
```

- Run the following command

```
java -Dspring.cloud.bootstrap.location=<properties file>  
-Dsolidify.oauth2.accesstoken=$OAUTH2_TOKEN -jar DLCM-Tools-2.2.7.jar >>dcm-tools.log 2>&1
```