20 Delft-FEWS as Command Line Runnable : Data Conversion Module - DCM

Introduction

The Data Conversion Module (DCM) is a generic application with which timeseries data can be converted. The DCM is a ‘stripped’ version of Delft FEWS and therefore supports the same import and export file formats. It is also possible to perform some transformations that are available in the transformation module of Delft-FEWS.

Basically the DCM ensures that (1) Delft-FEWS starts up, (2) that the relevant workflows are run and (3) that Delft-FEWS is shut off. The workflows contain tasks that ensure the import of the available import files, the conversion of these imported files and finally the export of the converted files to an export folder. One of the basic DCM-properties is the deletion of the Delft-FEWS database (or *localDataStore*) after each DCM run. This means that information from previous DCM runs is not available.

The DCM can be used in combination with the Data Interface Module (DiM). The interaction between the DiM and the DCM is illustrated in Figure 1.1. The DiM collects data from an external source and puts these files into a predefined import folder. This import folder contains a sub-folder for each datafeed. The DCM imports all files from this import folder and carries out the required conversions on the data. The results of these conversions are being exported in the desired file format to a predefined export folder. The DiM collects the data from this location and distributes it to the data destination.

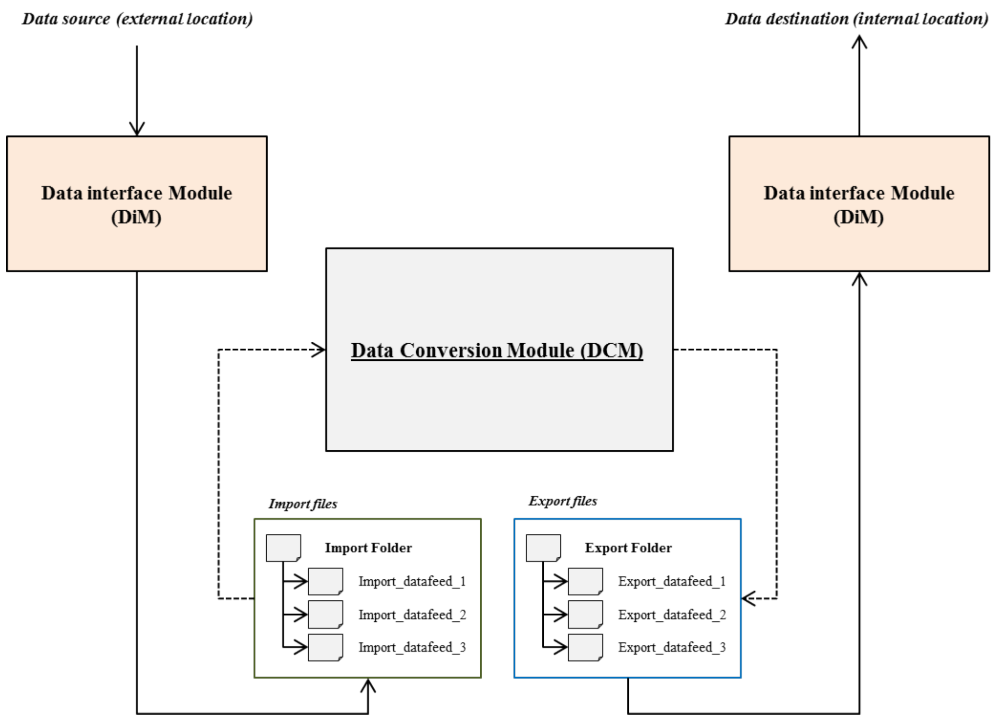


Figure 1: *Illustration of interaction between Data Interface Module (DiM) and Data Conversion Module (DCM).*

Binaries / installation

For FEWS versions 2018 and earlier The DCM had a separate bin folder and its own patches and was using the installed Java runtime version 1.8 (or earlier) These can still be requested through FEWS Support.

Starting with FEWS release 2019.02 rather then distributing a separate bin folder (which would need to include Java 11 runtime files) the choice was made to include DCM in the main FEWS distribution. The code of the data conversion module is now part of the main Delft_FEWS.jar and is run using the Delft-FEWSc.exe executable. The deliverable package for DCM that can be requested through FEWs support now contains a standalone configuration to use as a template for you project and a startup script (DataConversion.sh / DataConversion.bat) that you can use to start the DCM.

To install the DCM, the procedure is pretty much like any FEWS installation, the main difference difference is that after installing a base-build, you request from the same branch the data conversion package, with the file name dataconversion-stable-NNNN,NN.zip (where NNNN.NN is the FEWS version number) and unzip this as your region home folder. Then finally you get the latest FEWS patch for the given branch, however instead of putting the patch in the region home folder you need to put it in the main DCM folder and rename it to patch.jar.

Configuration

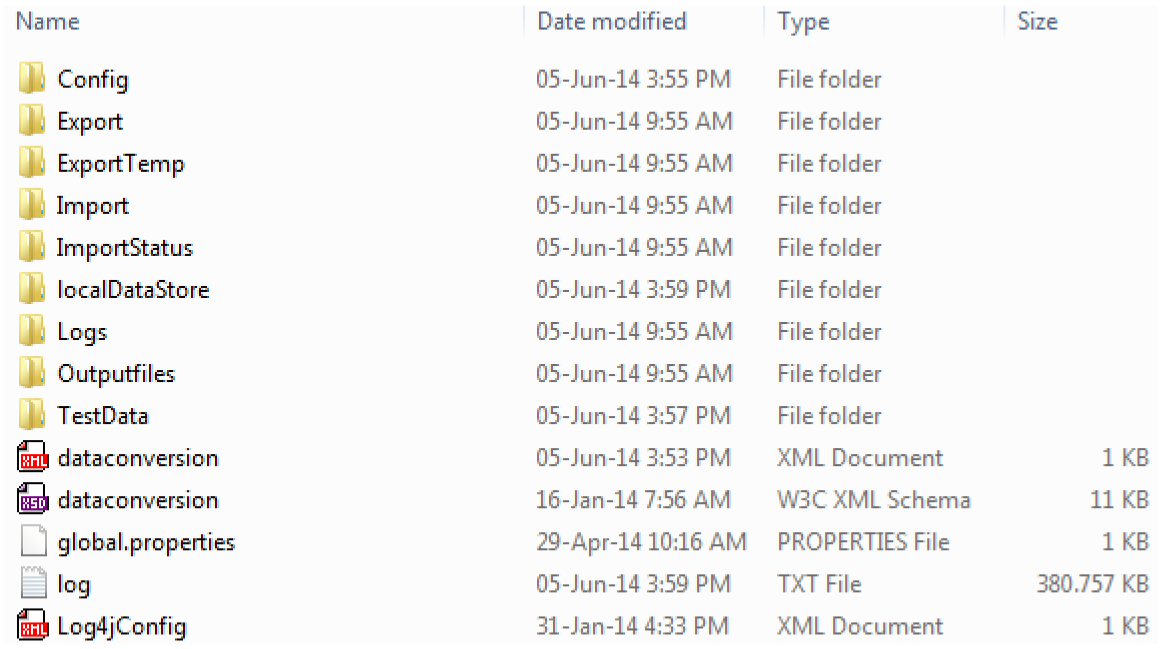
File structure of DCM after installation

Figure 2: *Overview of directory structure after installation of the DCM.*

After installation of the DCM your system will contain the following files (see Figure 2):

bin	Directory containing the FEWS distribution (2019.02 or newer)
patch.jar	latest patch file for this FEWS distribution
DataConversion_SA	FEWS Region Home directory for the DCM (contains the XML-configuration files)
DataConversion.bat	Script file needed to run DCM on a Windows machine
DataConversion.sh	Script file needed to run DCM on a Linux machine

This is the complete set of files needed to run the DCM, the FEWS . In the directory *DataConversion_SA* one can find all configuration files of the DCM (see Figure 3). The DCM configuration is based upon the Delft-FEWS configuration; this implies that its configuration is distributed over several configuration files. All configuration files are written in XML format.



Name	Date modified	Type	Size
Config	05-Jun-14 3:55 PM	File folder	
Export	05-Jun-14 9:55 AM	File folder	
ExportTemp	05-Jun-14 9:55 AM	File folder	
Import	05-Jun-14 9:55 AM	File folder	
ImportStatus	05-Jun-14 9:55 AM	File folder	
localDataStore	05-Jun-14 3:59 PM	File folder	
Logs	05-Jun-14 9:55 AM	File folder	
Outputfiles	05-Jun-14 9:55 AM	File folder	
TestData	05-Jun-14 3:57 PM	File folder	
dataconversion	05-Jun-14 3:53 PM	XML Document	1 KB
dataconversion	16-Jan-14 7:56 AM	W3C XML Schema	11 KB
global.properties	29-Apr-14 10:16 AM	PROPERTIES File	1 KB
log	05-Jun-14 3:59 PM	TXT File	380.757 KB
Log4jConfig	31-Jan-14 4:33 PM	XML Document	1 KB

Figure 3**:** *Example of DataConversion_SA directory. Note the dataconversion.xml in the root configuration.*

DCM configuration file

In Figure 3 the dataconversion file can be distinguished, this file is the DataConversion configuration file. The DCM configuration file contains all required instructions in order to run the DCM. This configuration file is an XML file that is structured according to the schema file ‘dataconversion.xsd’ which can be found in the release package.

```
    <complexType name="DataConversionComplexType">
        <sequence>
            <element name="clearOnStartup" type="boolean" default="true" minOccurs="0">
                <annotation>
                    <documentation>Control if local datastore is cleared on startup. By default eacht new run 
of the
                    DCM will start with an empty datastore.</documentation>
                </annotation>
            </element>
            <element name="activities" type="fews:ActivitiesComplexType"/>
        </sequence>
    </complexType>
    <!--WftrActivitiesComplexType -->
    <complexType name="ActivitiesComplexType">
        <sequence>
            <choice minOccurs="0" maxOccurs="unbounded">
                <element name="purgeActivity" type="fews:PurgeActivityComplexType">
                    <annotation>
                        <documentation>Purges a single file or a set of files within a directory.<
/documentation>
                    </annotation>
                </element>
                <element name="copyActivity" type="fews:CopyActivityComplexType">
                    <annotation>
                        <documentation>Copies single file or a set of files from source directory to a 
destination
                            directory. It is possible to add prefix or suffix to original file names.<
/documentation>
                    </annotation>
                </element>
                <element name="workflowActivity" type="fews:WorkflowActivityComplexType">
                    <annotation>
                        <documentation>Define FEWS workflow to run.</documentation>
                    </annotation>
                </element>
                <element name="moveActivity" type="fews:MoveActivityComplexType">
                    <annotation>
                        <documentation>Move single file or a set of files from source directory to a destination
                            directory.</documentation>
                    </annotation>
                </element>
                <element name="importStatusActivity" type="fews:ImportStatusActivityComplexType">
                    <annotation>
                        <documentation>Exports the import status to file. When running import in loop import 
status
                        must be run</documentation>
                    </annotation>
                </element>
                <element name="runInLoopActivityRunner" type="fews:RunInLoopActivityRunnerComplexType">
                    <annotation>
                        <documentation>Repeats the sub activities until their run method returns 'false'.<
/documentation>
                    </annotation>
                </element>
            </choice>
        </sequence>
    </complexType>
```

Figure 4: *DataConversion schema*

As of version 2015.02 the configuration file offers the option 'clearOnStartup' which makes the deletion of the local datastore at startup configurable.

The DataConversion configuration file contains a list of activities which will be run sequentially. There is no limit to the number and order of the activities. It is possible to choose from the following list of activities:

PurgeActivity:

With this activity it is possible to delete both files and directories. This activity supports the wildcards '?' and '*'.

This activity contains the following fields:

- id: required idendifier.
- description: optional information
- purgeFilePath: path to files that are to be deleted. Wildcards accepted.

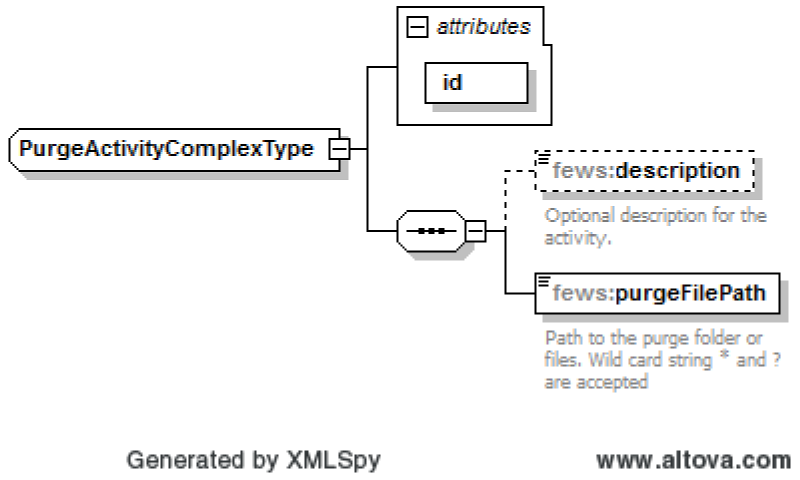

attributes
id
PurgeActivityComplexType
fews:description
Optional description for the activity.
fews:purgeFilePath
Path to the purge folder or files. Wild card string * and ? are accepted
Generated by XMLSpy
www.altova.com

Figure 5 *DataConversion purge activity*

CopyActivity:

With this activity it is possible to copy files from one location to another. This activity supports the wildcards '?' and '*'.

This activity contains the following fields:

- id: required idendifier.
- description: optional information
- srcFilePath: path to files that are to be copied. Wildcards accepted.
- destFilePath: path to destination directory. No wildcards allowed.
- prefix: optional text that can be added before the name of the destination file.
- suffix: optional text that can be added after the name of the destination file.
- setCurrentFileNameProperty: option that sets the name of the file currently being copied as a global property. This property can then be used in the configuration of other activities. This option only works when this activity is nested in a 'runInLoopActivityRunner' activity.

Example setCurrentFileNameProperty

```
<runInLoopActivityRunner id="runInLoop">
            <activities>

<!— This CopyActivity will copy all XML files form the Import/aquo directory. However because
The activity is being run in a loop each file will be copied individually. By setting the property
'setCurrentFileNameProperty' the name of the file being copied will be mapped to the key CURRENT_FILENAME. This
key can then be used by the other activities as shown below. -->

                <copyActivity id="loopedCopy">
                    <srcFilePath>%REGION_HOME%/TestData/Import/aquo/*.xml</srcFilePath>
                    <destFilePath>$IMPORT_FOLDER_ROOT$/aquo/</destFilePath>
                   <setCurrentFileNameProperty>CURRENT_FILENAME</setCurrentFileNameProperty>
                </copyActivity>
                <moveActivity id="loopedMove">
                   <srcFilePath>$EXPORT_FOLDER_ROOT $/aquo/*</srcFilePath>
                   <destFilePath>$BACKUP_FOLDER_ROOT$/aquo/$CURRENT_FILENAME$.xml</destFilePath>
                </moveActivity>
                ...
```

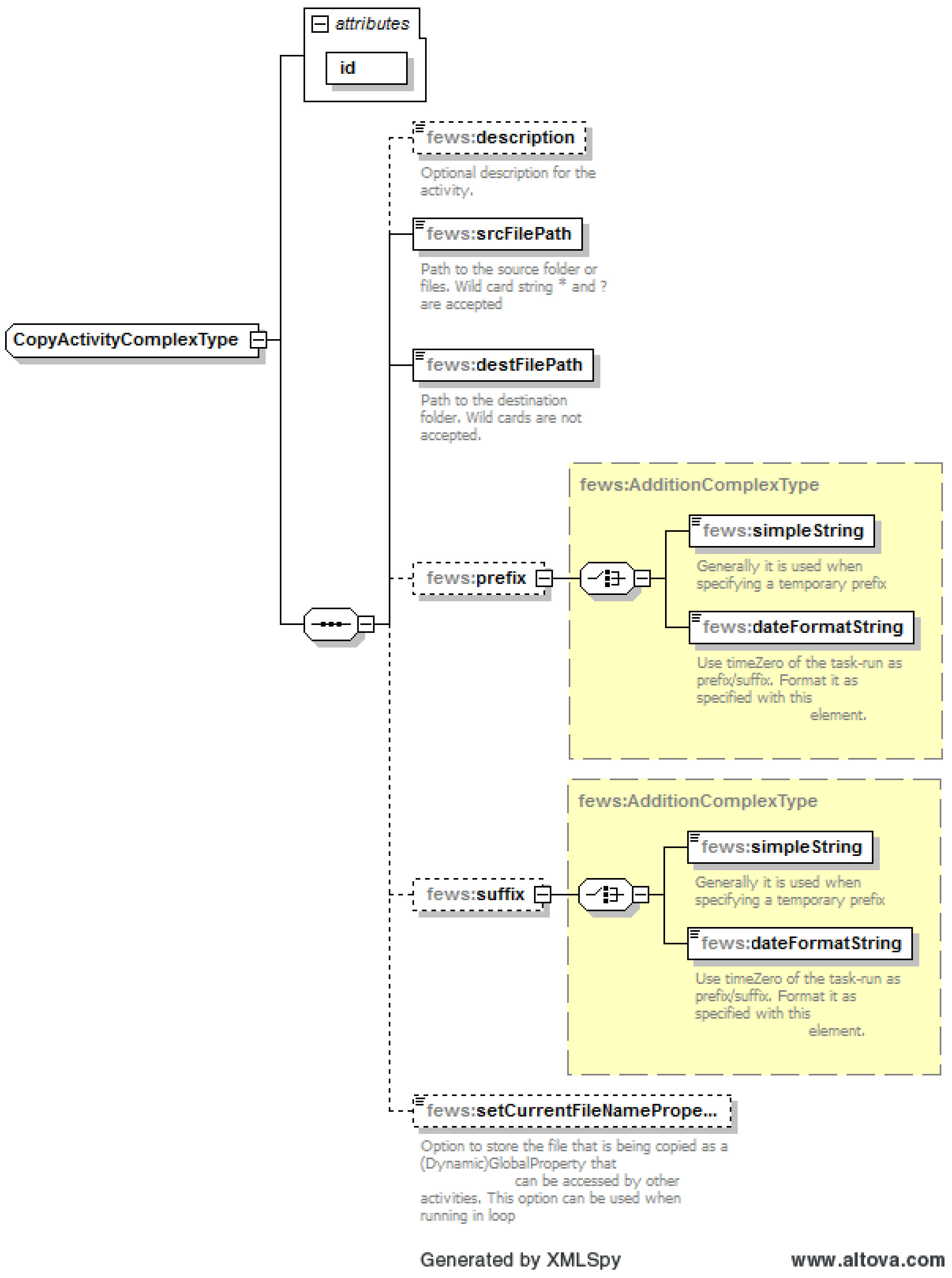

attributes
id
CopyActivityComplexType
fews:description
Optional description for the activity.
fews:srcFilePath
Path to the source folder or files. Wild card string * and ? are accepted
fews:destFilePath
Path to the destination folder. Wild cards are not accepted.
fews:prefix
fews:AdditionComplexType
fews:simpleString
Generally it is used when specifying a temporary prefix
fews:dateFormatString
Use timeZero of the task-run as prefix/suffix. Format it as specified with this element.
fews:suffix
fews:AdditionComplexType
fews:simpleString
Generally it is used when specifying a temporary prefix
fews:dateFormatString
Use timeZero of the task-run as prefix/suffix. Format it as specified with this element.
fews:setCurrentFileNamePrope...
Option to store the file that is being copied as a (Dynamic)GlobalProperty that can be accessed by other activities. This option can be used when running in loop
Generated by XMLSpy
www.altova.com

Figure 6 *DataConversion copy activity*

WorkflowActivity:

With this activity FEWS workflows can be run.

This activity contains the following fields:

- description: optional information
- workflowId: id of workflow to be run

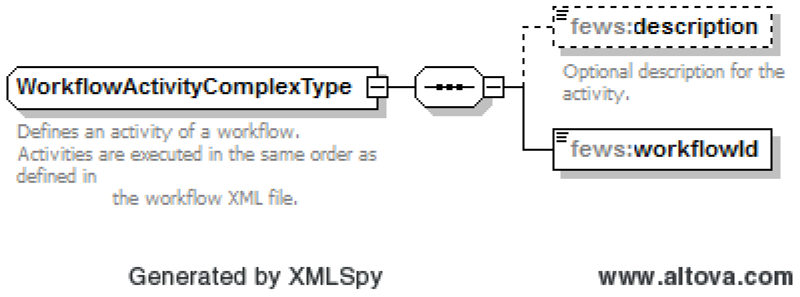

WorkflowActivityComplexType
Defines an activity of a workflow. Activities are executed in the same order as defined in the workflow XML file.
fews:description
Optional description for the activity.
fews:workflowId
Generated by XMLSpy
www.altova.com

Figure 7 *DataConversion workflow activity*

MoveActivity:

With this activity it is possible to move files from one location to another. This activity supports the wildcards '?' and '*'.

This activity contains the following fields:

- id: required idendifier.
- description: optional information
- srcFilePath: path to files that are to be copied. Wildcards accepted.
- destFilePath: path to destination directory. No wildcards allowed.
- setCurrentFileNameProperty: option that sets the name of the file currently being moved as a global property. This property can then be accessed by other activities. This option only works when this activity is nested in a 'runInLoopActivityRunner' activity.

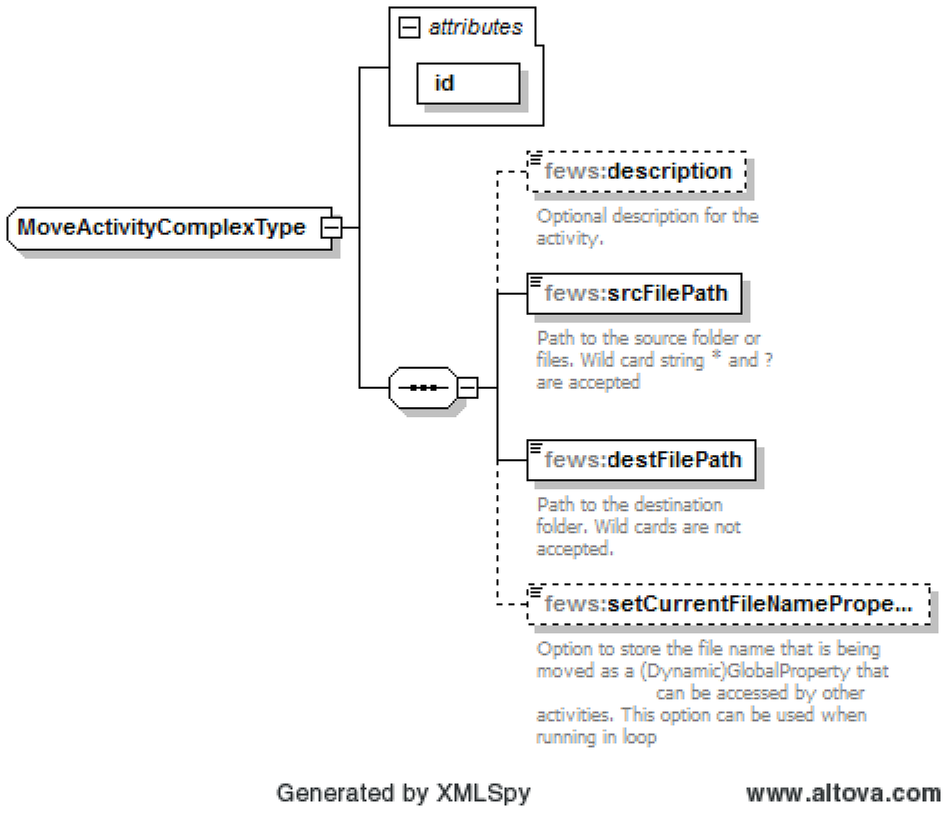

attributes
id
MoveActivityComplexType
fews:description
Optional description for the activity.
fews:srcFilePath
Path to the source folder or files. Wild card string * and ? are accepted
fews:destFilePath
Path to the destination folder. Wild cards are not accepted.
fews:setCurrentFileNamePrope...
Option to store the file name that is being moved as a (Dynamic)GlobalProperty that can be accessed by other activities. This option can be used when running in loop
Generated by XMLSpy
www.altova.com

Figure 8 *DataConversion move activity*

ImportStatusActivity:

With this activity the content of the ImportStatus table can be exported to CSV file. The ImportStatus is information produced by the Delft FEWS import modules, describing basic information about the last import run.

This activity contains the following fields:

- id: required idendifier.
- description: optional information
- importStatusPath: path to the directory in which status file is to be generated.
- dataFeedId: option to filter the 'dataFeedId' values that are to be exported to file. If omitted all available datafeed information will be exported.

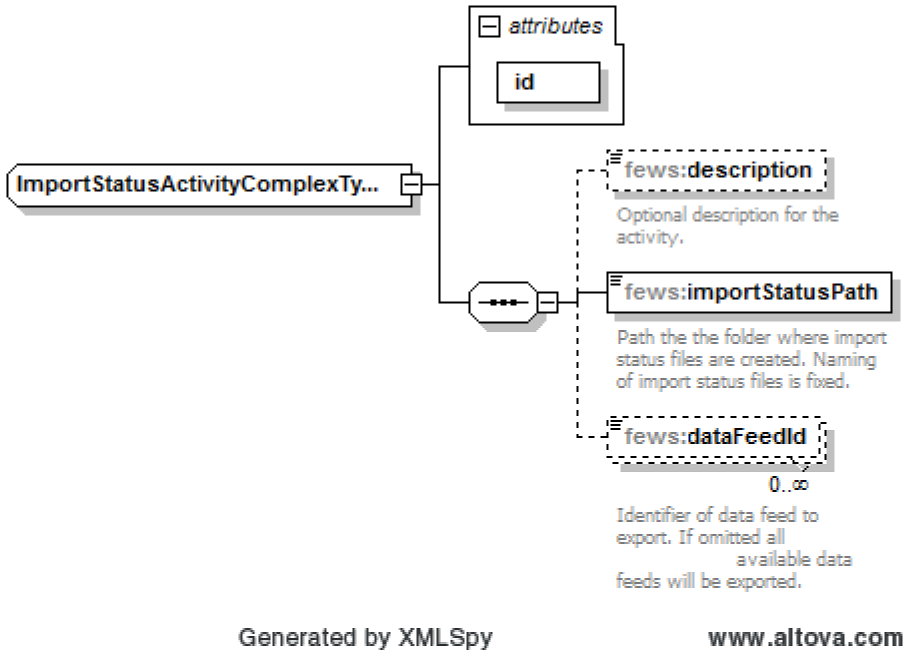

attributes
id
ImportStatusActivityComplexTy...
fews:description
Optional description for the activity.
fews:importStatusPath
Path the the folder where import status files are created. Naming of import status files is fixed.
fews:dataFeedId
0..∞
Identifier of data feed to export. If omitted all available data feeds will be exported.
Generated by XMLSpy
www.altova.com

Figure 9 *DataConversion import status*

RunInLoopActivity:

With this activity one or more of the above can be run repeatedly. This activity will keep looping over the sub activities until one of the sub activities indicates that it has completed all its loops or until the 'timeOut' has been exceeded.

This activity contains the following fields:

- id: required idendifier
- description: optional information
- activities: list of activities over which to loop
- timeOutSeconds: option timeout in seconds after which looping is terminated.

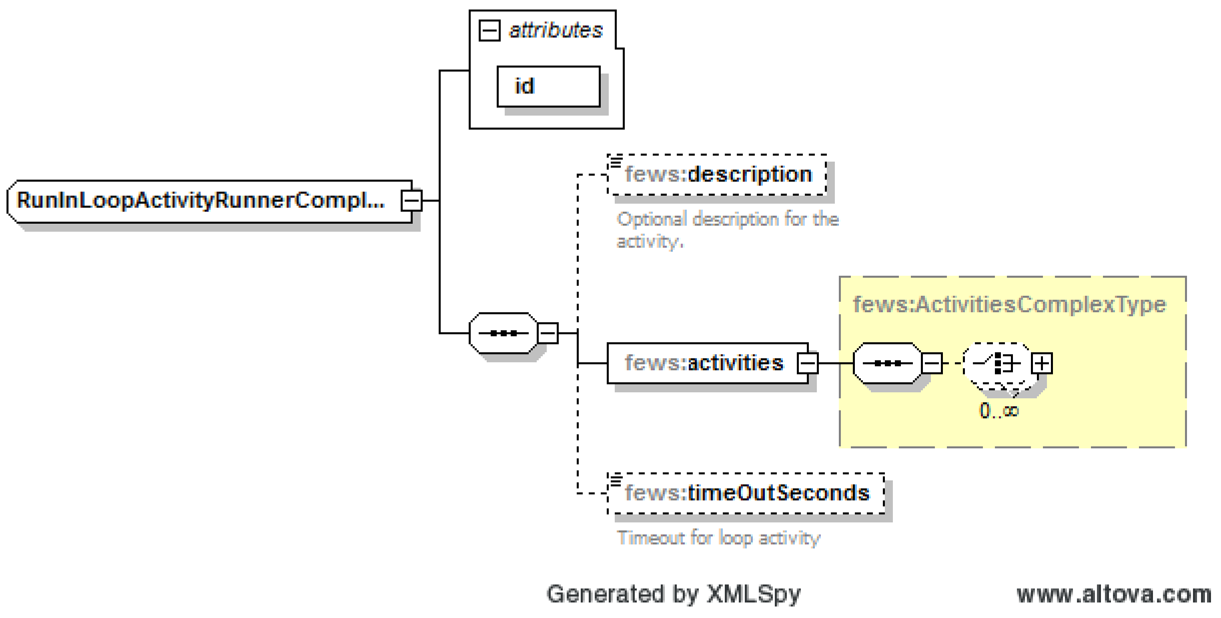

attributes
id
RunInLoopActivityRunnerCompl...
fews:description
Optional description for the activity.
fews:ActivitiesComplexType
fews:activities
0..∞
fews:timeOutSeconds
Timeout for loop activity
Generated by XMLSpy
www.altova.com

Figure 10 *DataConversion run in loop activity*

Configuration of DCM: conversion task variants

The DCM makes use of the file based Delft-FEWS configuration. Hence, all modules that are available in Delft-FEWS can also be used for the DCM. The configuration is located in the Config directory of the 'region_home' directory (see Figure 2). In this section some general remarks about the Delft-FEWs configuration in relation to the DCM can be found. We do this by defining three 'conversion tasks' with increasing complexity:

- Variant 1: workflow with import and export module.
- Variant 2: workflow with import, transformation (only on new data) and export module.
- Variant 3: workflow with import, transformation (on new data and on data of previous DCM runs) and export module .

Variant 1: Import, Export.

This is the simplest workflow variant. The workflow only consists out of an import and an export module. This variant can be used when the original import files already have the correct timestep and when location specific transformations are not needed. In this case the DCM is mainly used to convert the file format of the import files. In the Figure below a schematic illustration of this workflow variant is given.

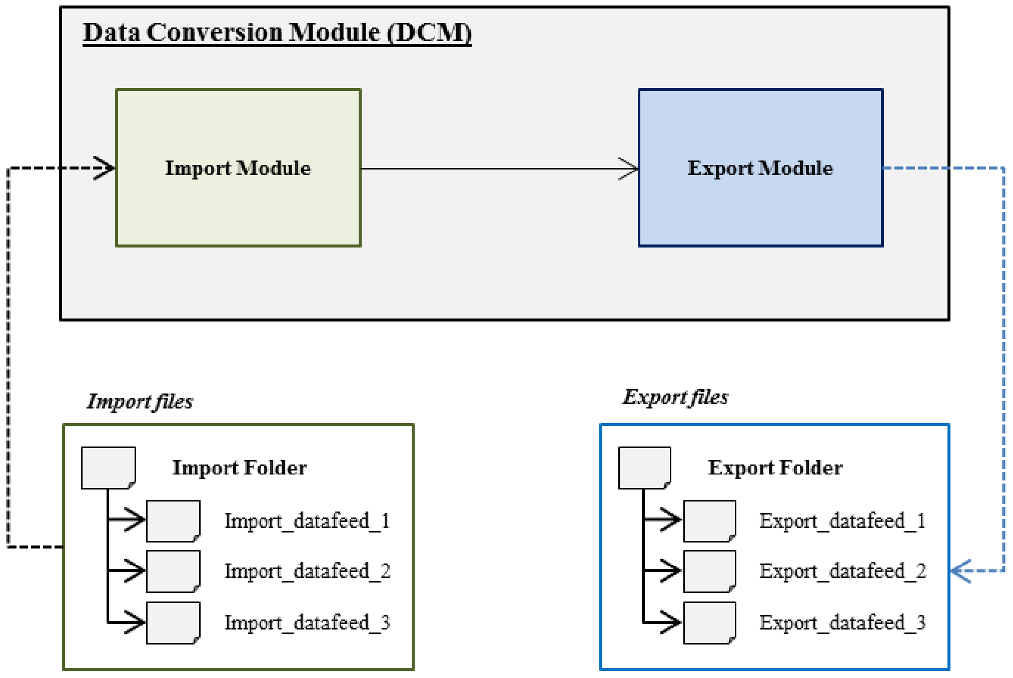


Figure 11 *Simplest workflow variant in DCM. The DCM is only used for conversion of the file format, there is no need for transformations.*

The DCM support the TimeSeriesImportRun module. This implies that all import types available within Delft-FEWS are also available to the DCM. To minimize the amount of configuration, it is possible to configure imports without having to specify the timeseries information in the import module. Here is an example for importing PI-Xml:

TimeSeriesImportRun in DCM

```
<?xml version="1.0" encoding="UTF-8"?>
<timeSeriesImportRun xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:
 schemaLocation="http://www.wldelft.nl/fews http://fews.wldelft.nl/schemas/version1.0/timeSeriesImportRun.xsd">
        <import>
                <general>
                        <importType>LmwWeatherdata</importType>
                        <folder>$IMPORT_FOLDER_ROOT$/LMW_maasaa</folder>
                        <idMapId>IdImport_LMWmaasaa</idMapId>
                </general>
                <temporary>true</temporary>
        </import>
</timeSeriesImportRun>
```

In the above example it can be seen that the configuration of TimeSeriesSets has been replaced by the element '*temporary*'. This way it is no longer necessary to configure locations and parameters in the RegionConfig directory. These are generated on-the-fly during the import process. All imported data is stored as temporary data and will only be available to other tasks that are being executed within the same workflow as the import module. However, if you want to make use of the locationSets than it is necessary to configure both the locations and the locationSets in the RegionConfig directory.

As is the case with the import run, the export no longer requires the configuration of TimeSeriesSets. The export run will automatically export all data imported in the previous import actions of the same workflow run.

TimeSeriesExportRun in DCM

```
<timeSeriesExportRun xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/timeSeriesExportRun.xsd">
        <export>
                <general>
                        <exportType>NETCDF-CF_TIMESERIES</exportType>
                        <folder>$EXPORT_FOLDER_ROOT$/LMW_maasaa</folder>
                        <exportFileName>
                                <name>_LMWmaasaa.nc</name>
                                <prefix>
                                        <timeZeroFormattingString>yyyy-MM-dd'T'HHmmss</timeZeroFormattingString>
                                </prefix>
                        </exportFileName>
                        <unitConversionsId>ExportUnitConversions</unitConversionsId>
                        <exportMissingValueString>-9999.0</exportMissingValueString>
                        <exportTimeZone>
                                <timeZoneOffset>+01:00</timeZoneOffset>
                        </exportTimeZone>
                </general>
                <properties>
                        <bool value="false" key="includeFlags"/>
                        <bool value="false" key="includeComments"/>
                </properties>
                <metadata>
                        <title>Export of LMW_maasaa datafeed</title>
                        <institution>Deltares</institution>
                </metadata>
        </export>
</timeSeriesExportRun>
```

Variant 2: Import, Transformation, Export.

The second workflow variant is similar to the first workflow variant. Here, the timestep of the original timeseries is also correct but additional transformations are required. A transformation module is added to the workflow to allow for location specific conversions (e.g. correction of water level) and /or the derivation of new timeseries. For the transformation module it remains necessary to explicitly configure TimeSeriesSet information. However it is not necessary to pre configure parameters and locations in the RegionConfig directory. As long as the required timeseries have been imported during a prior import run they will be available for the transformation.

In the Figure below a schematic illustration of workflow variant 2 is given:

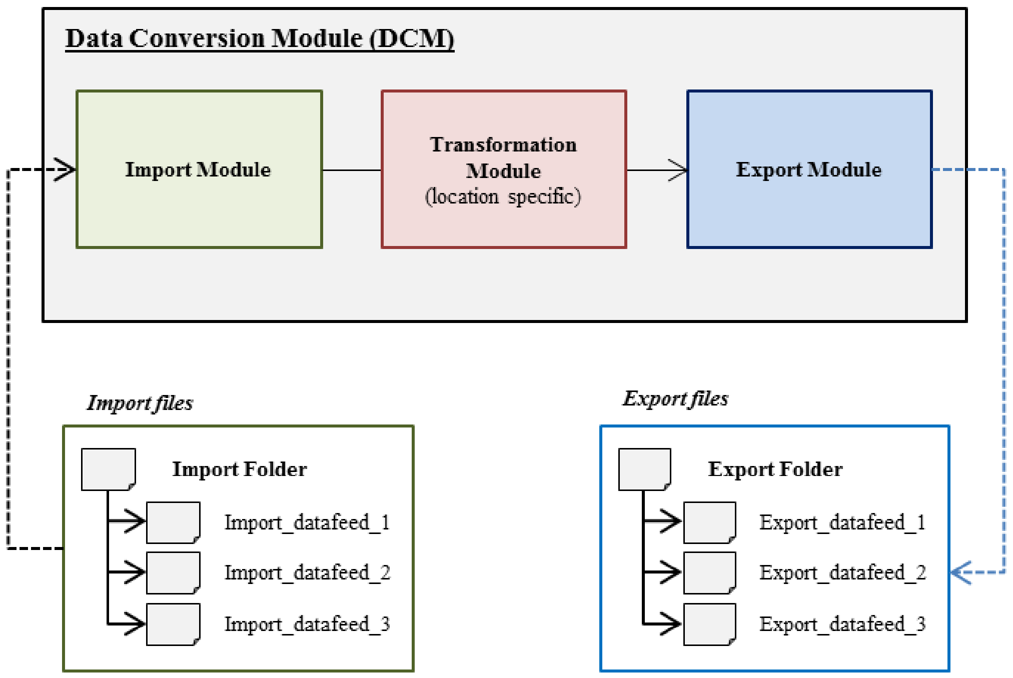


Figure 12 *Using Transformation functionality to change timeseries.*

It is possible to filter the data that is to be exported by configuring TimeSeriesSet information. But in that case the locations and parameter information must be added to the export configuration (see export example in Variant 2). Here is an example of a PI-export that includes a filter:

TimeSeriesExportRun in DCM with filter on export

```
<?xml version="1.0" encoding="UTF-8"?>
<timeSeriesExportRun xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/timeSeriesExportRun.xsd">
        <export>
                <general>
                        <exportType>PI</exportType>
                        <folder>$EXPORT_FOLDER_ROOT$/LMW_lixhe</folder>
                        <exportFileName>
                                <name>_LMWlixhe.txt</name>
                                <prefix>
                                        <timeZeroFormattingString>yyyy-MM-dd'T'HHmmss</timeZeroFormattingString>
                                </prefix>
                        </exportFileName>
                        <unitConversionsId>ExportUnitConversions</unitConversionsId>
                        <exportMissingValueString>-9999.0</exportMissingValueString>
                        <exportTimeZone>
                                <timeZoneOffset>+01:00</timeZoneOffset>
                        </exportTimeZone>
                </general>
                <properties>
                        <bool value="false" key="includeFlags"/>
                        <bool value="false" key="includeComments"/>
                </properties>
                <metadata>
                        <title>Export of LMW_lixhe datafeed</title>
                        <institution>Deltares</institution>
                </metadata>
                <timeSeriesSet>
                        <moduleInstanceId>LMWlixhe_interpolation</moduleInstanceId>
                        <valueType>scalar</valueType>
                        <parameterId>C10P</parameterId>
                        <locationSetId>LMWlixhe</locationSetId>
                        <timeSeriesType>temporary</timeSeriesType>
                        <timeStep unit="minute" multiplier="10"/>
                        <relativeViewPeriod unit="hour" start="-3" end="0"/>
                        <readWriteMode>read complete forecast</readWriteMode>
                </timeSeriesSet>
        </export>
</timeSeriesExportRun>
```

Variant 3: Import, (dis)Aggregation, Export.

The third workflow variant is used when the original import data does not have the appropriate timestep. In Delft-FEWS it is relatively simple to convert the original import data to the appropriate timestep by using a (dis-)aggregation function.

However, in order to perform a (dis-)aggregation it is necessary to have data for multiple timesteps. In an operational system the following aspects have to be taken into account:

- It is likely that external data is delivered per timestep. Hence, these files only contain data for one single timestep.
- One of the characteristics of the DCM is that it removes all information from its localDataStore when Delft-FEWS is closed. Consequently, import files from previous DCM runs are not available in the current DCM run.

This means that simply adding a (dis-)aggregation transformation will not work because there is only information for a single timestep. In order to prevent this problem several modules are added to the workflow which enable the 'storage' of information from previous timesteps.

In Figure 13 a schematic illustration of this third workflow variant is given. In this workflow an additional import from the back-up folder is added. The files from the back-up folder contain the import data from the previous DCM runs. In the second step of this workflow the files from both imports are merged. The merge transformation contains a hierarchy which states that the (new) import data is more important than the (old) back-up import data. The result of this merge transformation is a new timeseries which contains both new and old import data. Part of this new timeseries is exported to the back-up folder to serve as back-up timeseries in the next DCM run.

This approach guarantees the availability of a timeseries that contains data for multiple timesteps. It is now possible to carry out a (dis-)aggregation transformation to convert the import data to the appropriate timestep.

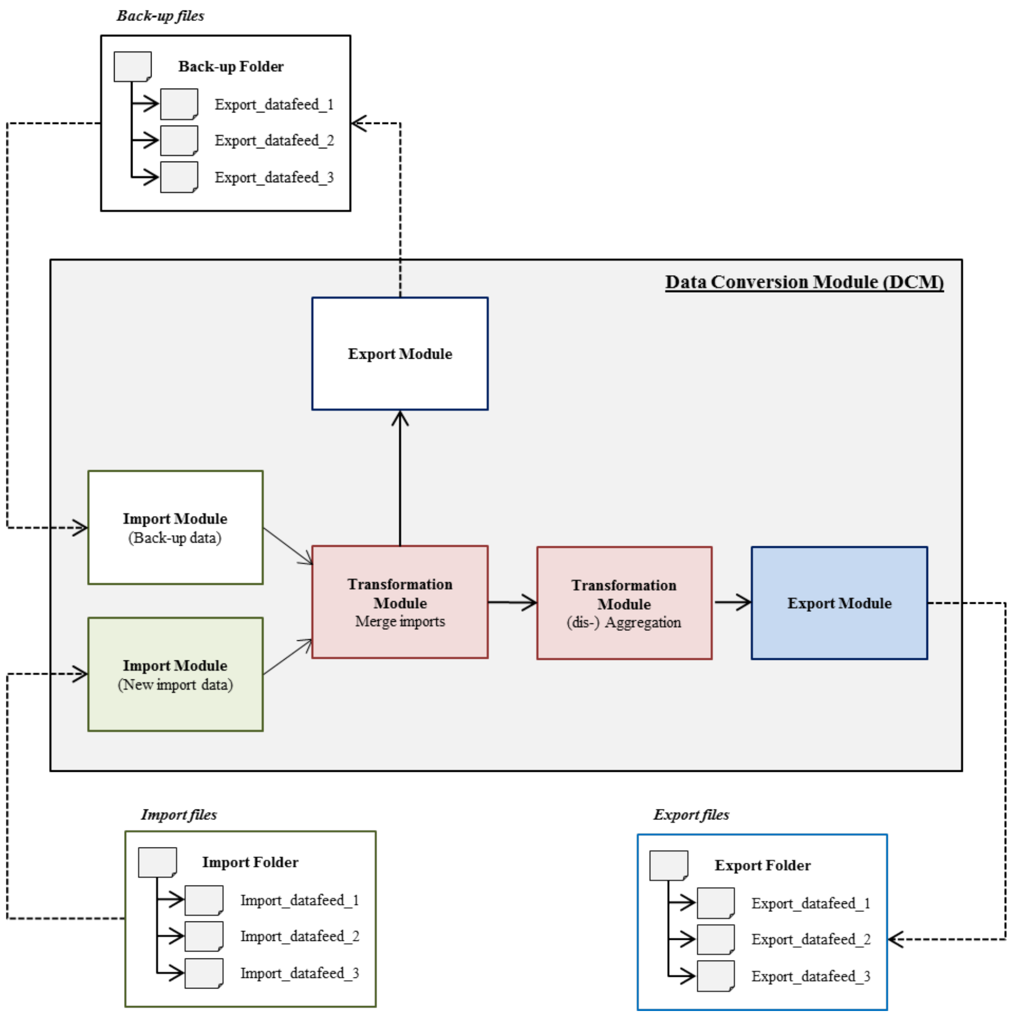


Figure 13 *This construction is needed whenever functionalities are used that need data from previous DCM runs. The extra export/import routine ensures that the user has data from previous DCM runs.*

Run the DCM

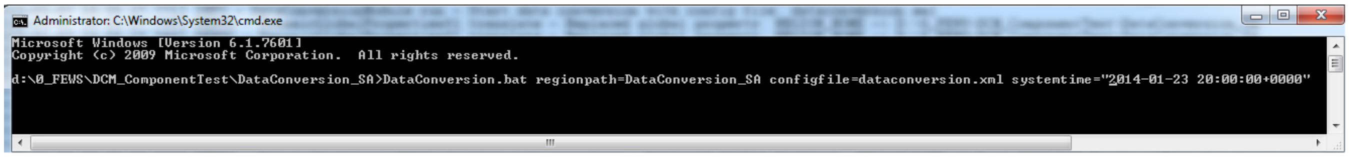
The DCM can be executed as a script from the command line or as a scheduled task (see Figure 2 for the location of these files).

- On a Windows machine the DCM is run using the BATCH file: **DataConversion.bat**
- On a Linux machine the DCM is run using the bash script file: **DataConversion.sh**

Both scripts require the same arguments:

- regionpath=<name or path> : Name of region directory containing all configuration files (DataConversion_SA). If the region directory is not located in the same directory as the script then a path is required.
- configfile=<name or path> : Name of the DCM configuration file (here we use the dataconversion.xml). If the DCM configuration file is not located in the root of the region directory then a path is required.
- workflowid=<identifier> : Identifier of workflow that is to be run. This argument is required when the 'configfile' is not provided. Can be comma separated array of workflow ids.
- importpath=<path> : Path to folder containing import files. The value of this argument will overrule global property IMPORT_FOLDER_ROOT. It is therefore necessary that this property is configured in the global.properties file. Furthermore this argument only works in combination with 'workflowid' argument.
- exportpath=<path> : Path to where export files will be written. The value of this argument will overrule global property EXPORT_FOLDER_ROOT. It is therefore necessary that this property is configured in the global.properties file. Furthermore this argument only works in combination with 'workflowid' argument.

- systemtime=<timestamp>   : Timestamp value (yyyy-MM-dd HH:mm:ssZ) to be used as System Time when running workflows. (note: Z for non-GMT time zones needs to be replaced by the timezone offset, i.e. +1000 for CET)
- binpath=<name or path>   : Path to the directory containing the Delft-FEWS binaries (bin). This argument is configured in the script files.
- clearonstart=<true or false>   : Option to turn off deletion of local datastore on startup. default is true.
- compact=<true or false>   : Option to compact the datastore on startup using a 'rolling barrel' algorithm. default is false.
- loglevel=<error, warn, info or debug> : option to set the log level, default is info. You can set it to 'debug' for troubleshooting, 'warn' or even 'error' to filter out non-critical information.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

d:\0_FEWS\DCM_ComponentTest\DataConversion_SA>DataConversion.bat regionpath=DataConversion_SA configfile=dataconversion.xml systemtime="2014-01-23 20:00:00+0000"
```

Figure 14: *Screenshot of the command prompt when running the DCM on a Windows machine.*