22 Locations and attributes defined in CSV files, Shape-DBF files or external tables

day for the validity of the location attributesdbServerType

Function:	_Functionality to define locations and to generate locationSets from a CSV file /Shape-DBF file / DB Table
Where to Use?	*Locations, LocationSets, IdMaps, DisplayGroups, ThresholdValueSets and ValidationRuleSets*
Why to Use?	*To have only one file or a set of files where all region specific information is stored.*
Description:	*Based on the CSV, DBF-Shape file or DB table you can easily manage the configuration*
Available since:	*DelftFEWS200803*

Contents

Overview

To be able to have only one file that manages all the regional information, Delft-FEWS offers the functionality to use a CSV, DBF or shape files that can be linked to the configuration. Locations and locationSets can be automatically generated and useful information as idMaps, thresholdvalues or validation values can be derived from these tables. It is also possible to link to one or more CSV/DBF files that contain time-dependent attributes. This can be used to define time-dependent coefficients that can be used by the new transformation module.

Finally you have a configuration that has many links to the CSV / DBF / shape files, but that will be managed only in these files. The advantage is that these files can simply be updated by automatic updating processes.

The functionality is based on the next principal:

- you generate locations in a locationSet and define *attributes* to these locations that store additional information like idMapping or validation limits.
- locationSets can be generated from the CSV, SHP/DBF, db table or from another locationSet by using conditions.
- idMaps can be linked to the location text attributes.
- all values in validationRuleSets can be linked to location number attributes
- all threshold values can be linked to location number attributes
- displayGroups can be generated automatically from locationSets
- it works both for regular point locations as for grids
- values in coefficientSetFunctions in a transformation config file can be linked to location number attributes

CSV file format

When the CSV file is not using the character set "Western Europe (ISO-8859-1)" the character set should be configured

DBF file format

This functionality only works for DBF files in the dBASE III format. When the Dbase file is not using the character set "Western Europe (ISO-8859-1)" the character set should be configured

Configuration

When there is no ID defined in the MapLayerFile (either csv, shape, dbf etc), the location is ignored. For example in the config csv example below, if on row 20 in the csv in column ID no value is defined, the location defined on row 20 will be ignored as a location for FEWS.

locationSets from CSV file

The most useful way is first to read all locations from the CSV file into one locationSet, where all attributes are assigned.
See for example:

```
<locationSet id="gegevensdump" editable="false">
    <csvFile>
      <file>gegevens.csv</file>
      <geoDatum>Rijks Driehoekstelsel</geoDatum>
      <id>%ID%</id>
      <name>%NAAM%</name>
      <description>%TYPE%</description>
      <iconName>%ICONFILE%</iconName>
      <toolTip><![CDATA[<html></html>]]></toolTip>
      <parentLocationId>%PARENT%</parentLocationId>
      <timeZoneOffset>+05:00</timeZoneOffset>
      <dateTimePattern>yyyyMMdd HH:mm:ss</dateTimePattern>
      <visibilityStartTime>%START%</visibilityStartTime>
      <visibilityEndTime>%EIND%</visibilityEndTime>
      <x>%X%</x>
      <y>%Y%</y>
      <z>0</z>
      <attribute id="PARENT">
        <text>%PARENT%</text>
      </attribute>
      <attribute id="TYPE">
        <text>%TYPE%</text>
      </attribute>
      <attribute id="CITECTLOC">
        <text>%CITECTLOC%</text>
      </attribute>
      <attribute id="IDMAP_Q">
        <text>%DEBIET%</text>
      </attribute>
      <attribute id="HMIN_Q">
        <number>%HMIN_Q%</number>
      </attribute>
      <attribute id="HMAX_Q">
        <number>%HMAX_Q%</number>
      </attribute>
      <attribute id="ROC_Q">
        <number>%ROC_Q%</number>
      </attribute>
      <attribute id="BIRTHDAY">
        <dateTime>%BIRTHDAY%</dateTime>
      </attribute>
    </csvFile>
  </locationSet>
```

Since 2018.01 attributes can be of type <dateTime>. To parse the text in the csv file to a valid date, the <dateTimePattern> and (optionally) <timeZoneOffset> fields are used. Please ensure these are set correctly.

Since 2021.02 you can configure ignoreIdsMissingInLocationSet=true when you expect the csv/dbf/table contains more locations than the location set. Don't use this function to attach the same csv file to multiple locations sets. This will increase the startup time of FEWS.

locationSets from esriShapeFile

The most useful way is first to read all locations from the Shape/DBF file into one locationSet, where all attributes are assigned.
See for example:

```
<locationSet id="gegevensdump" editable="false">
    <esriShapeFile>
      <file>gegevens</file>
      <geoDatum>Rijks Driehoekstelsel</geoDatum>
      <id>%ID%</id>
      <name>%NAAM%</name>
      <description>%TYPE%</description>
      <iconName>%ICONFILE%</iconName>
      <toolTip><![CDATA[<html></html>]]></toolTip>
      <parentLocationId>%PARENT%</parentLocationId>
      <timeZoneOffset>+05:00</timeZoneOffset>
      <dateTimePattern>yyyyMMdd HH:mm:ss</dateTimePattern>
      <visibilityStartTime>%START%</visibilityStartTime>
      <visibilityEndTime>%EIND%</visibilityEndTime>
      <x>%X%</x>
      <y>%Y%</y>
      <z>0</z>
      <attribute id="PARENT">
        <text>%PARENT%</text>
      </attribute>
      <attribute id="TYPE">
        <text>%TYPE%</text>
      </attribute>
      <attribute id="CITECTLOC">
        <text>%CITECTLOC%</text>
      </attribute>
      <attribute id="IDMAP_Q">
        <text>%DEBIET%</text>
      </attribute>
      <attribute id="HMIN_Q">
        <number>%HMIN_Q%</number>
      </attribute>
      <attribute id="HMAX_Q">
        <number>%HMAX_Q%</number>
      </attribute>
      <attribute id="ROC_Q">
        <number>%ROC_Q%</number>
      </attribute>
    </esriShapeFile>
  </locationSet>
```

locationSets from geoJSON file

```
<locationSet id="Provinces">
        <geoJsonFile>
                <file>provinces.geojson</file>
                <shapeType>polygon</shapeType>
                <id>%name%</id>
                <name>%name%</name>
                <attribute id="level">
                        <number>%level%</number>
                </attribute>
        </geoJsonFile>
</locationSet>
```

locationSets from database table

It is also possible to read locations directly from a database table. The contents of the database table are on the fly read and converted to a DBZ file. This DBZ file will be used by FEWS. This is for backup purpose in case the database is not available any more, like in stand-alone test environments.

```
<locationSet id="grondwater">
  <table>
        <databaseServer>
                <url>jdbc:oracle:thin:@dummy_hostname:1521:dummy_databasename</url>
                <user>dummy_username</user>
                <encryptedPassword>dummy_password_encrypted</encryptedPassword>
        </databaseServer>
    <name>PEILBUIZEN</name>
    <geoDatum>Rijks Driehoekstelsel</geoDatum>
    <id>ult_%FEWS_ID%</id>
    <name>%NAAM%</name>
    <description>Gebiedsnaam: %GEBIEDSNAA% </description>
    <x>%X_COORDINA%</x>
    <y>%Y_COORDINA%</y>
    <attribute id="DOMMEL_ID">
      <text>%MEETPUNTCO%</text>
    </attribute>
    <attribute id="TMX_ID">
      <text>%TMX%</text>
    </attribute>
    <attribute id="HARD_MAX">
      <number>%HARD_MAX%</number>
    </attribute>
    <attribute id="HARD_MIN">
      <number>%HARD_MIN%</number>
    </attribute>
  </table>
</locationSet>
```

In this example the above database table PEILBUIZEN is read from the database and completely converted to a zipped DBase file, named PEILBUIZEN.dbz. This file is used by FEWS to read all the required data.
To use a Firebird/Derby database file you should use the element <databaseFile> instead of <dbServerName>. Other connection strings:

- jdbc:postgresql://dummy_hostname:port/dummy_databasename
- jdbc:oracle:thin:@dummy_hostname:port/dummy_servicename
- jdbc:oracle:thin:@dummy_hostname:port:SID
- jdbc:oracle:thin:@TNSName
- jdbc:sqlserver://dummy_hostname:port/dummy_databasename

To encrypt the password, use F12 menu (clipboard - > encrypt password, available since 2016.01)

location relations

Relations between location relations can be used in attribute functions, transformations and time series sets. Currently the *one to many* relations can only be used for time series sets in the filters.xml.

```
<locationSet id="Stations">
        <esriShapeFile>
                <file>Stations</file>
                <geoDatum>WGS 1984</geoDatum>
                <id>%ID%</id>
                <name>%ID%</name>
                <x>%X%</x>
                <y>%Y%</y>
                <z>0</z>
                <attributeFile>
                        <csvFile>upstream</csvFile>
                        <id>%ID%</id>
                        <oneToManyRelation id="UPSTREAM">
                                <relatedLocationId>%RELATION%</relatedLocationId>
                        </oneToManyRelation>
                        <relation id="CATCHMENT">
                                <relatedLocationId>%CATCHMENT%</relatedLocationId>
                        </relation>
                </attributeFile>
        </esriShapeFile>
```

time dependent z (since 2022.01)

When using a separate attribute file the z can be time dependent. The z is used for converting a local datum to a global datum (water level).

```
<locationSet id="LocationSetWithTimeDependentAttributes">
        <csvFile>
                <file>TimeDependentLocationSet.csv</file>
                <id>%id%</id>
                <x>0</x>
                <y>0</y>
                <attributeFile>
                        <csvFile>TimeDependentLocationSetAttributes.csv</csvFile>
                        <id>%id%</id>
                        <startDateTime>%START%</startDateTime>
                        <endDateTime>%END%</endDateTime>
                        <z>%z%</z>
                        <attribute id="timeDependentAttribute">
                                <text>%timeDependentAttribute%</text>
                        </attribute>
                </attributeFile>
        </csvFile>
</locationSet>
```

```
ID;START;END;timeDependentAttribute;z
TD_Loc1;19000101;21000101;A;1
TD_Loc2;19000101;19850103;A;2
TD_Loc2;19850103;21000101;B;3
TD_Loc3;19000101;19850103;B;4
TD_Loc3;19850103;21000101;A;5
TD_Loc4;19000101;19850103;A;6
TD_Loc4;19850103;21000101;B;7
TD_Loc5;19000101;19850103;A;8
TD_Loc5;19850103;21000101;B;9
```

using attributes, constraints, relations etc.

In the above example the visibilityStartTime and visibilityEndTime tags are used to define the columns in the CSV/DBF file that contain the start and end dateTimes of the visibilityPeriod for each location. The (optional) visibilityPeriod is the period for which a location is visible in the user interface. The start and the end of the period are inclusive. Currently the visibility period is used in the map (explorer) window, the time series display and the spatial display. If startDateTime is not defined, then the location is visible for all times before endDateTime. If endDateTime is not defined, then the location is visible for all times after startDateTime. If startDateTime and endDateTime are both not defined, then the location is visible for all times. Furthermore the (optional) dateTimePattern tag is used to define the pattern for the dateTimes defined in the CSV/DBF file. If dateTimePattern is not specified, then the default pattern "yyyyMMdd" is used, which is the internal format that a CSV/DBF file uses for columns of type 'D' (date columns). The (optional) timeZoneOffset is the offset of the times in the CSV/DBF file, relative to GMT. For example "+02:00" means GMT+02:00. If no offset is specified, then time zone GMT is used by default.

Next you can derive the required locationSets from this dump by using constraints.
You can use constraints like:

- attributeTextEquals
- attributeTextContains
- attributeTextStartsWith
- idContains
- attributeExists
- etc (see schema or the schema diagram)

For example:

```
<locationSet id="ST_K.meting" editable="false">
    <locationSetId>gegevensdump</locationSetId>
    <constraints>
      <not>
        <attributeTextEquals equals="" id="IDMAP_KLEP"/>
      </not>
      <attributeTextEquals equals="Stuwen" id="TYPE"/>
    </constraints>
</locationSet>
```

It is also possible in a locationSet to link to time-dependent attributes. Time-dependent attributes need to be defined in a separate CSV/DBF file. In the locationSet use the attributeFile tag to make a reference to such a file. The following xml example has a reference to the file PumpStationsAttributes.dbf, which contains attributes that have different values for different periods in time, as well as different values for different locations. In this case the startDateTime and endDateTime tags are used to define the columns in the CSV/DBF file that contain the start and end dateTimes for each row. A given row in the CSV/DBF file contains values that are only valid between the time period for that row. This period is defined by the optional startDateTime and endDateTime for that row. If a row has no startDateTime, then it is valid always before the endDateTime. If a row has no endDateTime, then it is valid always after the startDateTime. If a row has no startDateTime and no endDateTime, then it is always valid. When Time-dependent attributes are change in time, make sure the startDateTime of one row is the same as the endDateTime of the previous row. If there is a gap of one day between the startDeteTime and the endDateTime, then there is a gap of one day for the validity of the location attributes. Since the 2021.01 the <attributeFile> can also be used to add additional attributes when the locations itself are read from a database table <table> .

```
<locationSet id="PumpStations">
  <esriShapeFile>
    <file>PumpStations</file>
    <geoDatum>WGS 1984</geoDatum>
    <id>%ID%</id>
    <name>%ID%</name>
    <x>%X%</x>
    <y>%Y%</y>
    <z>0</z>
    <attributeFile>
      <csvFile>PumpStationsAttributes.csv</csvFile>
      <id>%ID%</id>
      <timeZoneOffset>+05:00</timeZoneOffset>
      <dateTimePattern>dd-MM-yyyy HH:mm</dateTimePattern>
      <startDateTime>%START%</startDateTime>
      <endDateTime>%EIND%</endDateTime>
      <attribute id="speed">
        <number>%FREQ%</number>
      </attribute>
      <attribute id="discharge">
        <number>%POMPCAP%</number>
      </attribute>
    </attributeFile>
  </esriShapeFile>
</locationSet>
```

link shape file or geoJSON file to csv file

```
<locationSet id="Meteo Stations">
        <csvFile>
                <file>Meteo_Stations</file>
                <geoDatum>WGS 1984</geoDatum>
                <id>%ID%</id>
                <name>%ID%</name>
                <x>%X%</x>
                <y>%Y%</y>
                <z>0</z>
                <attribute id="region">
                        <description>Catchment</description>
                        <text>%REGION%</text>
                </attribute>
                <attributeFile>
                        <esriShapeFile>Meteo_Stations_Polygons</esriShapeFile>
                        <id>%ID%</id>
                        <attribute id="freq">
                                <number>%FREQ%</number>
                        </attribute>
                </attributeFile>
        </csvFile>
</locationSet>
```

locationIcons

It is possible to define the location icon with a new option in the locationSets derived from CSV/Shape-DBF files. You can define the location icon with the element iconName. The icon files should be defined as complete file name and this file should be available in the Config\IconFiles directory. If you want to refer to Config\IconFiles\Waterlevel.gif, you should define the iconName as

```
<iconName>Waterlevel.gif</iconName>
```

The old method by defining icons in the systemconfigfiles\locationIcons.xml is still available.

locations

The regional configuration file Locations is not needed any more, except for other locations that are not supplied in a CSV/DBF file.

attribute functions

Attribute functions are supported in configurations for parameter enumerations, id maps, primary validation, secondary validation, thresholds, transformation, general adapter, reports, map layer symbol sizes, statistics in time series dialog, display groups

@attributeId@

It is possible to reference an attribute that is available at a related location by prefixing an attribute id with a location relation id.

@relationId:attributeId@

examples

@CITECTLOC@_@IDMAP_DEBIET@

@discharge@ / 1000

@catchment:area@

idMaps

```
<locationIdFunction internalLocationSet="Meteo Stations" externalLocationFunction="@region@"/>
<locationIdPattern internalLocationSet="Pattern Stations" internalLocationPattern="H-*"
     externalLocationPattern="*"/>

..
or
..
  <function externalLocationFunction="@CITECTLOC@" internalParameter="Q.meting"
    externalParameterFunction="@IDMAP_DEBIET@" internalLocationSet="VV_Q.meting"/>
```

See that actual idMapping schema for all possible options.

Notice that you can use the location attributes as a function to map to the correct locations. You can create strings based on the attributes, like:

```
! uses the complete attribute value
externalParameterFunction="@IDMAP_DEBIET@"

! uses two concatenated attribute values
externalParameterFunction="@CITECTLOC@_@IDMAP_DEBIET@"

! uses attribute values concatenated with a fixed string
externalParameterFunction="@CITECTLOC@_DEBIET"
```

displayGroups

See all available options in the actual schema. The useful options for using together with the CSV/DBF configuration are explained here. Both options automatically generate the list of the locations in the shortcut trees. The list of locations is ordered alphabetically.

singleLocationDisplays
Adds multiple displays at once to this display group. Every display will show only one location.

singleParentLocationDisplays
Adds multiple displays at once to this display group. Every display will show only children for one parent location, and the parent location itself when specified in the time series sets.

```
<displayGroup name="Meteo">
    <singleParentLocationDisplays>
      <locationSetId>VV_P.meting.dag</locationSetId>
      <locationSetId>VV_P.meting</locationSetId>
      <parentLocationSetId>VV_P.meting.dag</parentLocationSetId>
      <parentLocationSetId>VV_P.meting</parentLocationSetId>
      <plotId>meteo</plotId>
    </singleParentLocationDisplays>
  </displayGroup>
```

Thresholds

you can use now ...Function alternatives for all the values

```
<levelThresholdValue>
        <levelThresholdId>LevelWarn</levelThresholdId>
        <description>.....</description>
        <valueFunction>@SOFT_MAX@</valueFunction>
        <upActionLogEventTypeId>TE.571</upActionLogEventTypeId>
</levelThresholdValue>
```

ValidationRuleSets

you can use now ...Function alternatives for all the values, like

- extremeValuesFunctions
- sameReadingFunctions
- etc...

```
<levelThresholdValue>
        <levelThresholdId>LevelWarn</levelThresholdId>
        <description>.....</description>
        <valueFunction>@SOFT_MAX@</valueFunction>
        <upActionLogEventTypeId>TE.571</upActionLogEventTypeId>
</levelThresholdValue>
```

CoefficientSetFunctions

In the new transformation module it is possible to define transformations with embedded coefficientSetFunctions in a transformation config file. For a given transformation, e.g. StructurePumpFixedDischarge, there is a choice between a coefficientSetFunctions object and a coefficientSet object. The coefficientSetFunctions object is the same as its corresponding coefficientSet counterpart, only all elements with a value are replaced by elements with a function. A function is a function expression that can refer to location attributes, e.g. '' (discharge) / 60". See the following xml example.

```
<transformation id="pump with coefficient set functions">
  <structure>
    <pumpFixedDischarge>
      ...
      <coefficientSetFunctions>
        <discharge>@discharge@ / 1000</discharge>
      </coefficientSetFunctions>
      ...
    </pumpFixedDischarge>
  </structure>
</transformation>
```

CoefficientSetFunctions are currently (as of build 30246) supported for the following transformations: userSimple, stageDischargePower, dischargeStagePower, filterLowPass and all structure transformations. See the pages of those specific transformations for configuration examples.

different types of attributes

When using coefficientSetFunctions, note that the elements can have different types, e.g. float, boolean, enumeration. For each coefficientSetFunctions type see the schema definition of its corresponding coefficientSet counterpart for the types of the different elements. The type of an attribute as defined in the locationSets configuration file must match the type of the element in which the attribute is used.

- For elements of type float (e.g. userSimple coefficient value) the attribute should be defined as a **number** attribute in the locationSets configuration file as follows:

```
<attribute id="coef_a">
  <number>%COEF_A%</number>
</attribute>
```

- For elements of type string, boolean (e.g. structureCrumpWeir energyHeadCorrection) or enumeration (e.g. stageDischarge type) the attribute should be defined as a **text** attribute in the locationSets configuration file as follows:

```
<attribute id="ehcorr">
  <text>%EHCORR%</text>
</attribute>
```

Coefficients that depend on location and time

A coefficientSetFunction can be very useful when using coefficients that depend on location and/or time. In that case the coefficientSetFunction needs to be defined only once with a link to the correct attributes. The attributes are defined in a CSV/DBF file. Then a transformation run will use the coefficientSetFunction to create coefficientSets for each location and time-period by taking the required values from the attributes from the CSV/DBF file automatically.

time-dependent attributes

If several attributes are used in the same coefficientSetFunction, then it is still possible to have some of those attributes time-independent and some time-dependent. However all the *time-dependent* attributes that are used in a given coefficientSet should be defined with exactly the same time-periods in the CSV/DBF file.

Coefficients with multiple values (tables)

Some transformations require a table, e.g. a head-discharge table, in a coefficientSet. For the purpose of tables it is possible to define a given attribute in a CSV/DBF file with multiple values. To do this make multiple rows with the same location and same period, only with different values for the attributes. If a given attribute is used in a table in a coefficientSetFunctions object, then for each location and period the multiple values that are defined for that location and period will be converted to a table during a transformation run. This only works for elements in a coefficientSetFunctions object that are designated as table elements. An element in a coefficientSetFunctions object is designated as a table element if, according to the schema, the element can occur only once in the coefficientSetFunctions object, but can occur multiple times in the corresponding coefficientSet object. This is how a transformation run knows that it should search for multiple values for attributes to create a table. This is the case e.g. for the headDischargeTableRecord element in the StructurePumpHeadDischargeTable transformation, which would be used as in the following xml example. In this case the "head" and "discharge" attributes should have multiple values defined in the CSV/DBF file so that a head-discharge table can be created.

> **tables**
>
> All attributes, e.g. "head" and "discharge", that are used in the same table element, e.g. headDischargeTableRecord, should have the same number of values defined per location per period. It is still possible to have a different number of values for different periods and different locations, as long as there are as many head values as discharge values per location per period.

```
<transformation id="pump with head-discharge table with coefficient set functions">
  <structure>
    <pumpHeadDischargeTable>
      ...
      <coefficientSetFunctions>
        <headDischargeTableRecord head="@head@" discharge="@discharge@ * 1000"/>
      </coefficientSetFunctions>
      ...
    </pumpHeadDischargeTable>
  </structure>
</transformation>
```

Sample input and output

Example of attribute file

```
<locationSet id="locations">
  <csvFile>
    <file>locations.csv</file>
    <id>%ID%</id>
    <x>%X%</x>
    <y>%Y%</y>
    <attribute ...../// first here your normal attributes
    </attribute>
    <attributeFile>
      <csvFile>pumpcurves.csv</csvFile>
      <id>%LOCID%</id>
      <attribute id="head">
        <number>%dH%</number>
      </attribute>
      <attribute id="discharge">
        <number>%Q%</number>
      </attribute>
    </attributeFile>
  </csvFile>
</locationSet>
```

example csv file

```
LOCID,dH,Q
0001,0,0
0001,0.1,1
0001,0.2,1.9
0001,0.3,2.5
0001,0.5,3.5
0001,1,5
```

Error and warning messages

Description of errors and warnings that may be generated

Error:	*exceeds 255 characters *.csv Illegal argument Column name* Error can appear when using standard encoding (ISO-8859-1) in the CSV file.
Action:	Reduce amount of characters in cell. Error will also appear when the column is not used in the definition of the LocationSet and/or attributes.

Known issues

Describe all known issues and unexpected behaviour

- Since 2016.01 an error is logged when values contain decimal comma instead of decimal point. e.g. 453,73 should be 453.73
- When using standard encoding (ISO-8859-1) Delft-FEWS will return an Illegal argument error when a Column name exceeds 255 characters.

Related modules and documentation

Links to related parts of the system

Technical reference

Entry in moduleDescriptors:	*Specification of: ENTRY and DESCRIPTION in the SystemConfigFiles\ModuleDescriptors.xml*

```
<moduleDescriptor id="ENTRY">
<description>DESCRIPTION</description>
```