

SmartMet Server

QUICK SETUP

Finnish Meteorological Institute
(2022-01-14)

1	QUICK SETUP	3
1.1	Purpose.....	3
1.2	Requirements.....	3
1.3	Phases	3
	PHASE 1: Parameter recognition and mapping	3
	PHASE 2: Filling the Content Storage.....	5
	STEP 1:.....	5
	STEP 2:.....	5
	STEP 3:.....	5
	STEP 4:.....	5
	STEP 5:.....	6
	PHASE 3: Checking grid information	7
	PHASE 4: Querying grid information	9
	PROBLEM SOLVING	10
	STEP 1:.....	10
	STEP 2:.....	10
	STEP 3:.....	10
	STEP 4:.....	11
	STEP 5:.....	12
	STEP 6:.....	13
	STEP 7:.....	13

1 QUICK SETUP

1.1 Purpose

The purpose of this document is to describe a quick setup for the SmartMet Server so that it can use grid-files (GRIB1, GRIB2, NetCDF, QueryData) for its timeseries queries.

1.2 Requirements

In this quick setup we expect that the following requirements are fulfilled before the setup is started:

1. The latest versions of the SmartMet components and configuration files are installed.
2. Redis server is running and accessible (locally or remotely) and its address and port is known.
3. The current filesystem contains a directory that contains valid grid files.

1.3 Phases

This quick setup is divided into the following phases:

1. Parameter recognition and mapping
2. Filling the Content Storage
3. Checking grid information
4. Querying grid information

PHASE 1: Parameter recognition and mapping

The first thing to do is to make sure that the system can recognize parameters in our grid files. The easiest way to test this is to use "**grid_dump**" application, which prints the content of the given grid file.

The current application needs to know the location of the main configuration file used by the "smartmet-library-grid-files" module (which is responsible for extraction of grid files). This location is given in the environment variable "**SMARTMET_GRID_CONFIG_FILE**". You can define this by the following command.

```
export SMARTMET_GRID_CONFIG_FILE=/usr/share/smartmet/grid-files/grid-files.conf
```

After that you should be able to execute the "**grid_dump**" application (this is usually located in "**/usr/bin/files**" directory). The application takes a grid file name as a parameter.

```
grid_dump /grib/ECG_20210309T000000_RH-PRCNT_pressure.grib
```

The result looks something like this:

```
-----
FILE : /grib/ECG_20210309T000000_RH-PRCNT_pressure.grib
-----
PhysicalGridFile
GridFile
- fileName      = /grib/ECG_20210309T000000_RH-PRCNT_pressure.grib
- fileId        = 0
- deletionTime  = 19700101T000000
- groupFlags    = 0
- producerId    = 0
- generationId  = 0
- numberOfMessages = 1

##### MESSAGE [0] #####

- filePosition  = 0 (0x0)
- fileType      = 1
- referenceTime = 20210309T000000
```

```

- forecastTime           = 20210309T000000
- gridProjection         = LatLon
- gridGeometryId       = 1007
- gridRowCount          = 1801
- gridColumnCount       = 3600
- fmiParameterId      = 163
- fmiParameterName    = RH-PRCNT
- fmiParameterUnits     = %
- fmiParameterLevelId   = 2
- parameterLevel        = 1000
- gribParameterId       = 157
- gribParameterName     = r
- gribParameterUnits    = %
- gribParameterDescription = Relative humidity
- newbaseParameterId    = 13
- newbaseParameterName  = Humidity
- netCdfParameterName   =
- gridHash              = 15139342594299081125

```

The recognition was successful if the “**fmiParameterId**”, “**fmiParameterName**” and “**gridGeometryId**” fields are not empty. If these fields are empty, but the “gribParameterId” is not empty then the current parameter was recognized as a GRIB-parameter, but this parameter was not mapped into the FMI parameter identifier. This mapping is required, because all parameter information is stored into the Content Database (= Redis database) with the FMI identifiers.

If the “**gridGeometryId**” is zero or empty then the application did not recognize the geometry used by the current grid. In this case, the application prints on screen the line that should be added into the geometry definition file (usually named as “**fmi_geometries.csv**”).

```

** GRIB1 Geometry not configured: /grib/ECG_20210309T000000_RH-PRCNT_pressure.grib
** Add the following line into the geometry definition file (=> fill id, name and description fields):

1;id;name;3600;1801;0.000000;90.000000;0.100000;0.100000;+x-y;description

```

The point of this identifier is that we might have the same data in multiple precisions and geometries. When we query this information, we should have a way to separate these geometries from each other. This is why we need unique geometry identifiers for different geometries.

If the parameter recognition failed, you should probably try to find a simple GRIB-file that contains some well-known parameters (like Temperature, Pressure, etc.) and try again. The default configuration of the current library should be able to recognize most common parameters used in GRIB files.

It is possible to configure this library so that it can recognize also very exotic parameters. This is a little bit complex operation and we do not explain it here. You can find details related to this operation in “**grid-files.pdf**” document.

If you cannot get the parameter recognition and mapping to work, then there is no reason to continue the setup before this is fixed.

PHASE 2: Filling the Content Storage

Now we know that the system is able to recognize and map parameters in grid files. The next step to do is to store this information into the Content Storage (= Redis database). The Content Storage is needed when this information is searched. It knows what kind of information is available and where it can be found.

We use the “**filesys2smartmet**” application in order to scan directories and searching grid files. When it finds a grid file, it recognizes grid parameters and all essential information (timesteps, levels, etc.) and stores this information into the Content Storage. Notice, that this information does not contain the actual grid data.

The current application can be executed like this:

```
filesys2smartmet <configFile> <loopWaitTimeInSeconds>
```

The first parameter is the name of the configuration file and the second parameter is the wait time (in seconds) between scanning loops. If the wait time is zero then the scanning is executed only once. The configuration file contains all necessary configuration information needed by the current application. Details of this configuration file is described in the “filesys2smartmet.pdf” document. In this document we describe required configuration steps in very simplified way.

STEP 1:

Define the location (= directories) of your grid files and patterns that are used for searching these files.

```
content-source :  
{  
  directories = [ "/grib"  
  patterns = [ "*.grib", "*.grib1", "*.grib2"  
}
```

STEP 2:

Define the producer information into the file, which name is defined with “producerDefFile” parameter. This producer information will be copied into the Content Storage.

```
ECG;ECG;ECMWF; European Centre for Medium-Range Weather Forecasts  
PAL;PAL_SCANDINAVIA; PAL-Scandinavia; PAL-Scandinavia Ground Level forecasts
```

STEP 3:

Rename your grid files so that they contain the producer’s short name and the generation/analysis time in the beginning of the file (separated by the ‘_’ character).

```
ECG_20220124T000000_filexxx.grib  
PAL_20220124T000000_filezdf3.grib
```

STEP 4:

Define the type and the location of the Content Storage (= Redis database):

```
content-storage : {  
  type = "redis"  
  redis :  
  {  
    address = "127.0.0.1"  
    port = 6379  
    tablePrefix = "a."  
  }  
}
```

STEP 5:

Execute the “**filesys2smartmet**” application:

```
filesys2smartmet /cfg/filesys-to-smartmet.conf 0
```

If everything went well, we should now have some information in the Content Storage. Notice that if we remove a grid file from the filesystem and execute the “**filesys2smartmet**” application again, then all information related to this file is removed from the Content Storage.

The easiest way to access content information manually is to use command-line client programs. There is a command-line client program for each service method defined in the Content Server API. For example, we can fetch a list of content information records from the Redis database by the following command:

```
cs_getContentList 0 0 0 100 -redis 127.0.0.1 6379 "a."
```

And the response would be something like this:

```
ContentInfoList
ContentInfo
- mFileId          = 2
- mFileType        = 2
- mMessageIndex    = 0
- mFilePosition    = 0
- mMessageSize     = 1816619
- mProducerId      = 1
- mGenerationId    = 1036
- mForecastTime    = 20220106T000000
- mForecastTimeUTC = 1641427200
- mFmiParameterId  = 162
- mFmiParameterName = TD-K
- mFmiParameterLevelId = 6
- mParameterLevel  = 2
- mForecastType    = 3
- mForecastNumber  = 4
- mFlags           = 0
- mSourceId        = 100
- mGeometryId      = 1087
- mModificationTime = 20220103T021143
- mDeletionTime    = 20220106T120000
```

You can fetch a list of all producers by the following command:

```
cs_getProducerInfoList 0 -redis 127.0.0.1 6379 "a."
```

You can fetch a list of all generations by the following command:

```
cs_getGenerationInfoList 0 -redis 127.0.0.1 6379 "a."
```

You can fetch a list of files belonging to the producer “ECG” by the following command:

```
cs_getFileInfoListByProducerName 0 ECG 0 100 -redis 127.0.0.1 6379 "a."
```

Notice that there are tens of other commands that you can use.

PHASE 3: Checking grid information

Now we know that we have content information available in the Content Storage. The next step is to start the SmartMet server so that we can get more detailed look to the current grid information.

In this phase we need just 1) the **grid-engine** and 2) the **grid-admin** and 3) the **grid-gui** plugins.

Both the grid-engine and the grid-admin need connection to the Content Storage (= Redis database). Define these connections into their main configuration files.

```
content-storage :
{
  type = "redis"

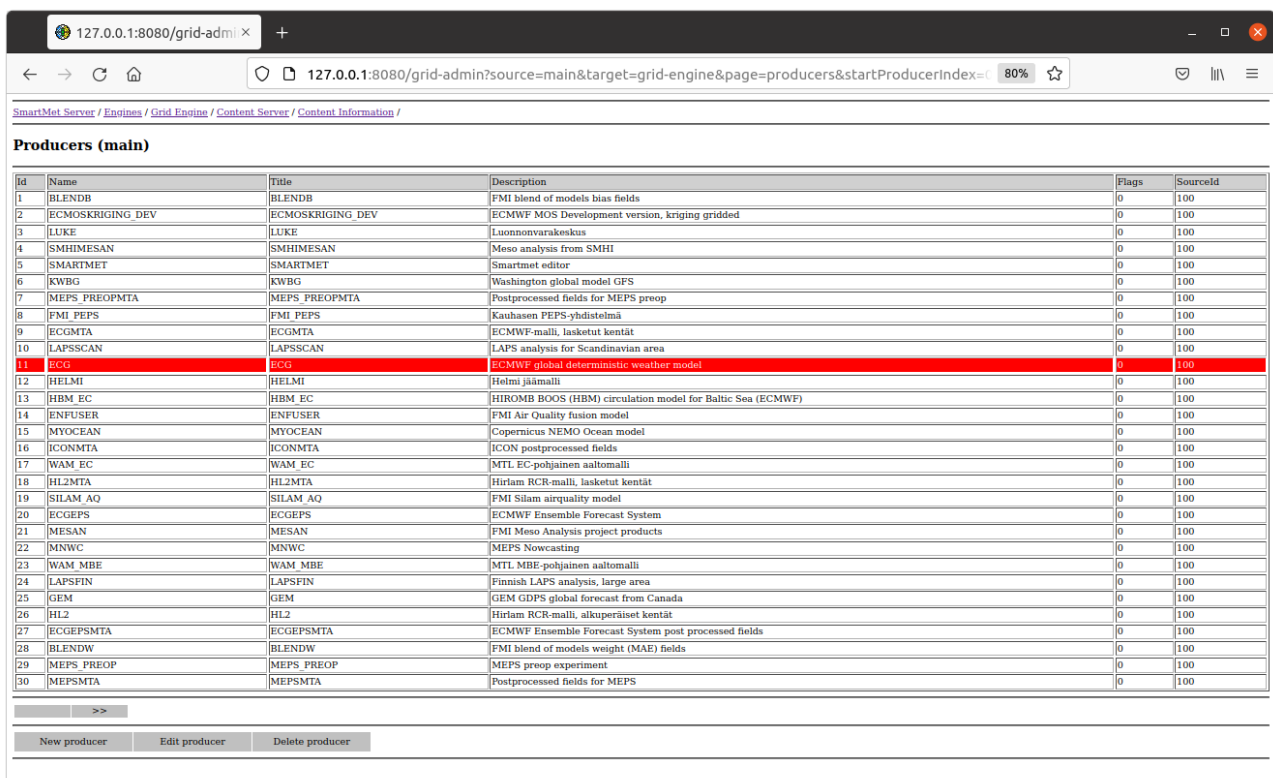
  redis :
  {
    address = "127.0.0.1"
    port    = 6379
    tablePrefix = "a."
  }
}
```

In this phase, we can ignore all other configuration parameters.

After the starting of the SmartMet Server we should be able to use the grid-admin and the grid-gui plugins.

The grid-admin plugin can be used for example examining information in the Content Storage. This is the information that the “**filesys2smartmet**” application saved into the Redis database.

You should use the admin-plugin in order to check that the Content Storage contains correct information related to producers, generations, files and content parameters.

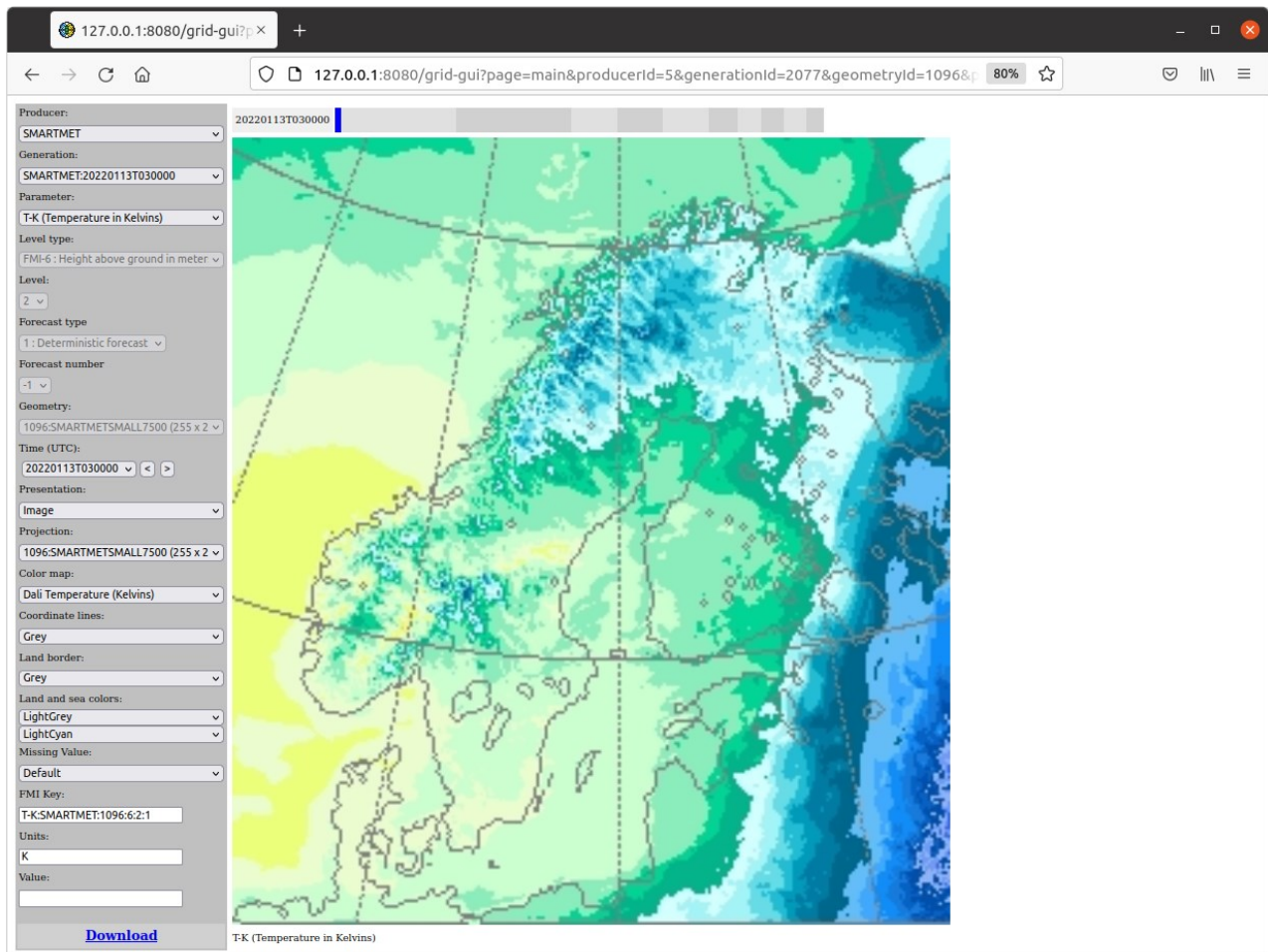


The screenshot shows a web browser window with the URL `127.0.0.1:8080/grid-admin`. The page title is "SmartMet Server / Engines / Grid Engine / Content Server / Content Information /". The main heading is "Producers (main)". Below it is a table with 6 columns: Id, Name, Title, Description, Flags, and SourceId. The table lists 30 producers, with the 11th row (Id 11, Name ECG, Title ECG) highlighted in red. At the bottom of the table, there are three buttons: "New producer", "Edit producer", and "Delete producer".

Id	Name	Title	Description	Flags	SourceId
1	BLENDDB	BLENDDB	FMI blend of models bias fields	0	100
2	ECMOSKRIGING_DEV	ECMOSKRIGING_DEV	ECMWF MOS Development version, kriging gridded	0	100
3	LUKE	LUKE	Luonnonvarakeskus	0	100
4	SMHIMESAN	SMHIMESAN	Meso analysis from SMHI	0	100
5	SMARTMET	SMARTMET	Smartmet editor	0	100
6	KWBG	KWBG	Washington global model GFS	0	100
7	MEPS_PREOPMTA	MEPS_PREOPMTA	Postprocessed fields for MEPS preop	0	100
8	FMI PEPS	FMI PEPS	Kanhasen PEPS-yhdistelmä	0	100
9	ECGMTA	ECGMTA	ECMWF-malli, lasketut kentät	0	100
10	LAPSSCAN	LAPSSCAN	LAPS analysis for Scandinavian area	0	100
11	ECG	ECG	ECMWF global deterministic weather model	0	100
12	HELM	HELM	Helmi jäätö	0	100
13	HBM EC	HBM EC	HIROMB BOOS (HBM) circulation model for Baltic Sea (ECMWF)	0	100
14	ENFUSER	ENFUSER	FMI Air Quality fusion model	0	100
15	MYOCEAN	MYOCEAN	Copernicus NEMO Ocean model	0	100
16	ICONMTA	ICONMTA	ICON postprocessed fields	0	100
17	WAM EC	WAM EC	MTL EC-pohjainen aaltomalli	0	100
18	HL2MTA	HL2MTA	Hiriam RCR-malli, lasketut kentät	0	100
19	SILAM AQ	SILAM AQ	FMI Silam airquality model	0	100
20	ECGEPS	ECGEPS	ECMWF Ensemble Forecast System	0	100
21	MESAN	MESAN	FMI Meso Analysis project products	0	100
22	MNWC	MNWC	MEPS Nowcasting	0	100
23	WAM MBE	WAM MBE	MTL MBE-pohjainen aaltomalli	0	100
24	LAPSPIN	LAPSPIN	Finnish LAPS analysis, large area	0	100
25	GEM	GEM	GEM GDPS global forecast from Canada	0	100
26	HL2	HL2	Hiriam RCR-malli, alkuperäiset kentät	0	100
27	ECGEPSMTA	ECGEPSMTA	ECMWF Ensemble Forecast System post processed fields	0	100
28	BLENDW	BLENDW	FMI blend of models weight (MAE) fields	0	100
29	MEPS PREOP	MEPS PREOP	MEPS preop experiment	0	100
30	MEPSMTA	MEPSMTA	Postprocessed fields for MEPS	0	100

The grid-gui plugin can be used to examine the actual grid data. The grid-plugin gets information from the available grids from the Content Storage via the grid-engine, which has cached this information into the memory and this way the information fetching is much faster than fetching the same information from the Redis database.

The grid-gui is able to visualize all kind of grid files. This is very easy way to check weather the grid data looks correct.



If the grid-admin and the grid-gui shows information properly then we have passed the most critical steps of the setup. Now we know that the system is able to 1) recognize information in grid files, 2) to store this information into the Content Storage and 3) to fetch this information from the Content Storage.

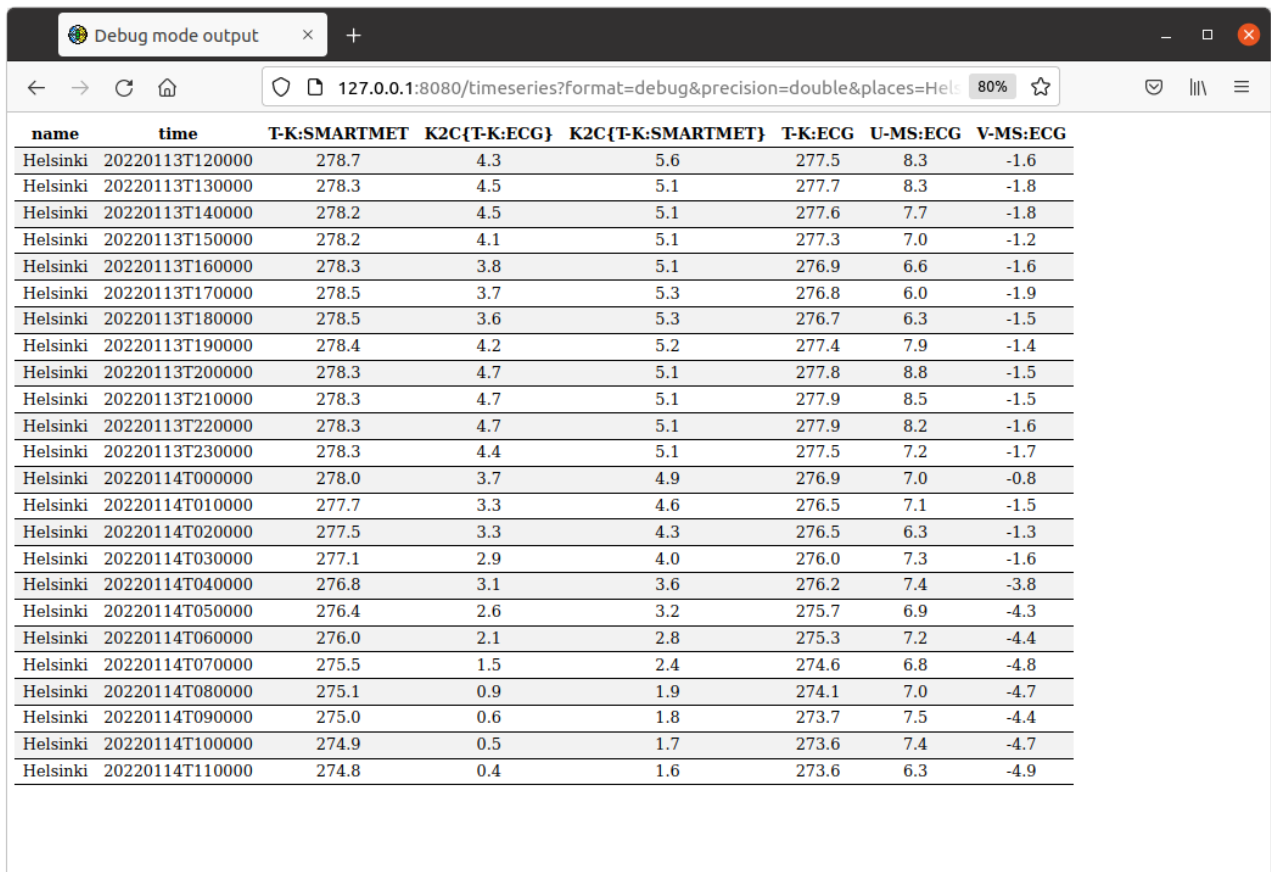
PHASE 4: Querying grid information

The next step is to make sure that we are able to query grid information via other plugins like Timeseries. First thing to do is to add the gis-engine, the geonames-engine, the querydata-engine and the timeseries-plugin into the SmartMet Server's main configuration file.

After that we should add available producers and geometries into the grid-engine's producer file (usually named as "producers.csv"). This file defines two things: 1) producer & geometry combinations that can be queried and 2) the search order of the producers if the producer is not given in the query.

```
SMARTMET;1096;;
SMARTMETMTA;1096;;
ECG;1007;;
ECGMTA;1007;;
ECG;1008;;
ECGMTA;1008;;
```

After that you should be able to use the timeseries plugin for querying grid parameters. If you want to be sure that you are querying the correct parameters (i.e. correct levels and geometries) then you should use the exact "FMI key", which is show in the grid-gui. Otherwise, this information is filled with default values found from different mapping files.



name	time	T-K:SMARTMET	K2C{T-K:ECG}	K2C{T-K:SMARTMET}	T-K:ECG	U-MS:ECG	V-MS:ECG
Helsinki	20220113T120000	278.7	4.3	5.6	277.5	8.3	-1.6
Helsinki	20220113T130000	278.3	4.5	5.1	277.7	8.3	-1.8
Helsinki	20220113T140000	278.2	4.5	5.1	277.6	7.7	-1.8
Helsinki	20220113T150000	278.2	4.1	5.1	277.3	7.0	-1.2
Helsinki	20220113T160000	278.3	3.8	5.1	276.9	6.6	-1.6
Helsinki	20220113T170000	278.5	3.7	5.3	276.8	6.0	-1.9
Helsinki	20220113T180000	278.5	3.6	5.3	276.7	6.3	-1.5
Helsinki	20220113T190000	278.4	4.2	5.2	277.4	7.9	-1.4
Helsinki	20220113T200000	278.3	4.7	5.1	277.8	8.8	-1.5
Helsinki	20220113T210000	278.3	4.7	5.1	277.9	8.5	-1.5
Helsinki	20220113T220000	278.3	4.7	5.1	277.9	8.2	-1.6
Helsinki	20220113T230000	278.3	4.4	5.1	277.5	7.2	-1.7
Helsinki	20220114T000000	278.0	3.7	4.9	276.9	7.0	-0.8
Helsinki	20220114T010000	277.7	3.3	4.6	276.5	7.1	-1.5
Helsinki	20220114T020000	277.5	3.3	4.3	276.5	6.3	-1.3
Helsinki	20220114T030000	277.1	2.9	4.0	276.0	7.3	-1.6
Helsinki	20220114T040000	276.8	3.1	3.6	276.2	7.4	-3.8
Helsinki	20220114T050000	276.4	2.6	3.2	275.7	6.9	-4.3
Helsinki	20220114T060000	276.0	2.1	2.8	275.3	7.2	-4.4
Helsinki	20220114T070000	275.5	1.5	2.4	274.6	6.8	-4.8
Helsinki	20220114T080000	275.1	0.9	1.9	274.1	7.0	-4.7
Helsinki	20220114T090000	275.0	0.6	1.8	273.7	7.5	-4.4
Helsinki	20220114T100000	274.9	0.5	1.7	273.6	7.4	-4.7
Helsinki	20220114T110000	274.8	0.4	1.6	273.6	6.3	-4.9

PROBLEM SOLVING

If the timeseries queries do not work immediately, then there is probably something wrong in the grid-engine's configuration. Our previous phases have approved that the current data is available and accessible, so the problem is most likely in the configuration parameters that are used in the Query Server. More detailed description of this configuration can be found from the "grid-engine.pdf" document.

STEP 1:

The grid-engine automatically generates mapping information for the parameters that it finds from the Content Storage. These mappings are generated to the following files:

```
query-server :
{
  mappingTargetKeyType = 2
  mappingUpdateFile :
  {
    fmi    = "%(DIR)/mapping_fmi_auto.csv"
    newbase = "%(DIR)/mapping_newbase_auto.csv"
    netCdf = "%(DIR)/mapping_netCdf_auto.csv"
  }
}
```

Make sure that these automatically generated files exist. If not, check the filesystem permissions.

STEP 2:

Usually, mappings in these automatically generated files are manually checked and copied into more permanent mapping files (for example, mapping_fmi_auto.csv => mapping_fmi.csv). That's because these automatically generated files are continuously overwritten. When mappings are copied then they are automatically removed from these automatically generated files.

Make sure that all permanent mapping files and also all automatically generated files are in the grid-engine's mapping file list.

```
query-server :
{
  mappingFiles =
  [
    "%(DIR)/mapping_fmi.csv",
    "%(DIR)/mapping_fmi_auto.csv",
    "%(DIR)/mapping_newbase.csv",
    "%(DIR)/mapping_newbase_auto.csv",
    "%(DIR)/mapping_netCdf.csv",
    "%(DIR)/mapping_netCdf_auto.csv"
  ];
}
```

STEP 3:

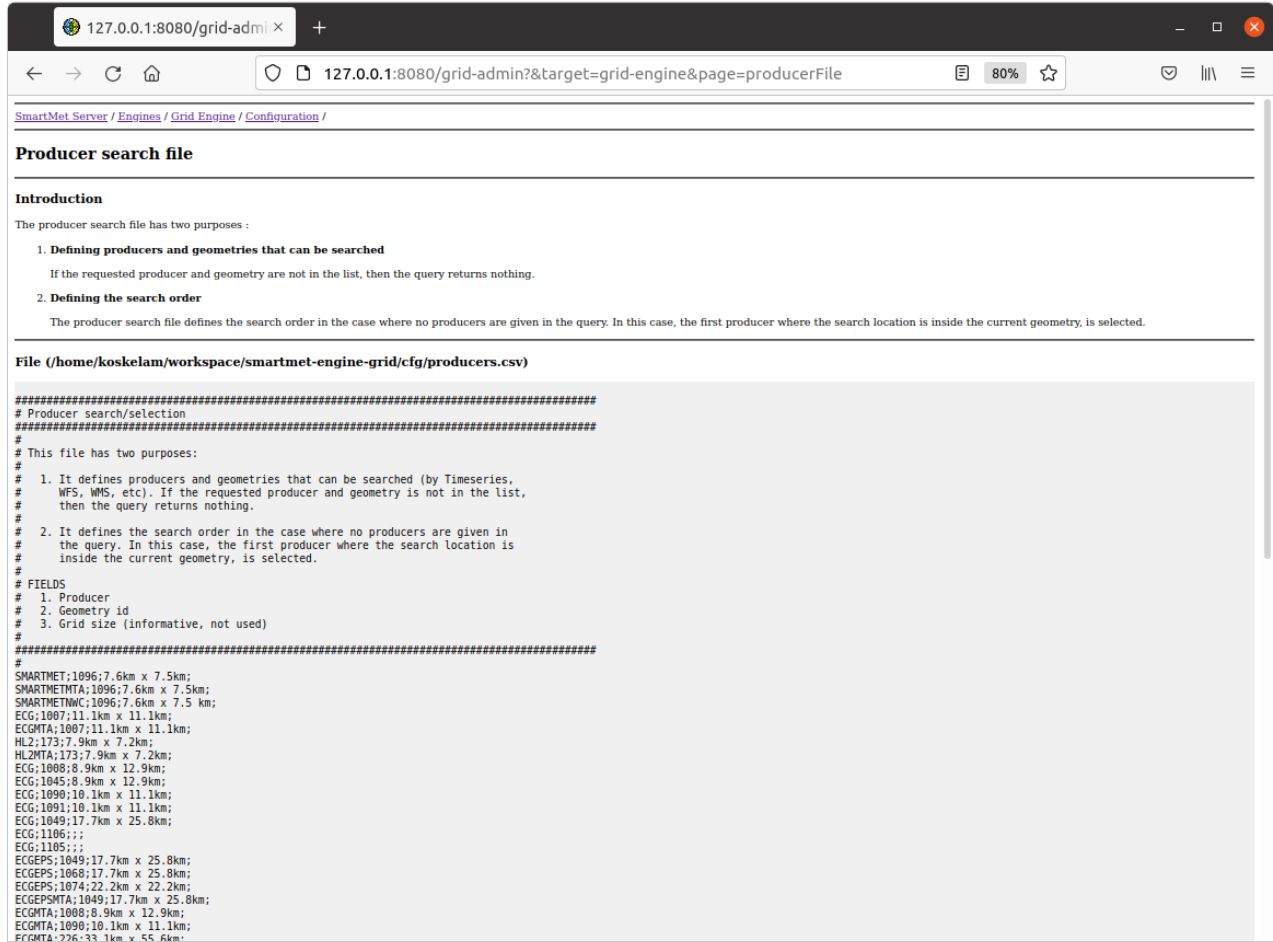
Check the content of the "mapping_fmi*.csv" files. You should find mapping definitions for the producers and parameters that you added earlier.

```
ECG;ALBEDO-OTO1;2;ALBEDO-OTO1;1007;1;1;00000;1;1;1;0;E;;;
ECG;ALBEDOSLR-OTO1;2;ALBEDOSLR-OTO1;1008;1;1;00000;1;1;1;0;E;;;
ECG;CAPE-JKG;2;CAPE-JKG;1007;1;1;00000;1;1;1;0;E;;;
```

Make sure that the geometry identifiers of these mappings are same as the geometry identifiers in the producer file that was defined in the previous phase.

STEP 4:

Make sure that you and the grid engine has the same understanding of the configuration. Probably the most common mistake is that you are edited configuration files that the grid-engine cannot even see. The easiest way to browse the configuration that the grid-engine is to use the grid-admin plugin. In this way you can see the content and the location of all configuration files that the grid-engine is using. For example, the producer search file that we defined in the previous phase might look like this.



127.0.0.1:8080/grid-admin

127.0.0.1:8080/grid-admin?&target=grid-engine&page=producerFile

SmartMet Server / Engines / Grid Engine / Configuration /

Producer search file

Introduction

The producer search file has two purposes :

- 1. Defining producers and geometries that can be searched**
If the requested producer and geometry are not in the list, then the query returns nothing.
- 2. Defining the search order**
The producer search file defines the search order in the case where no producers are given in the query. In this case, the first producer where the search location is inside the current geometry, is selected.

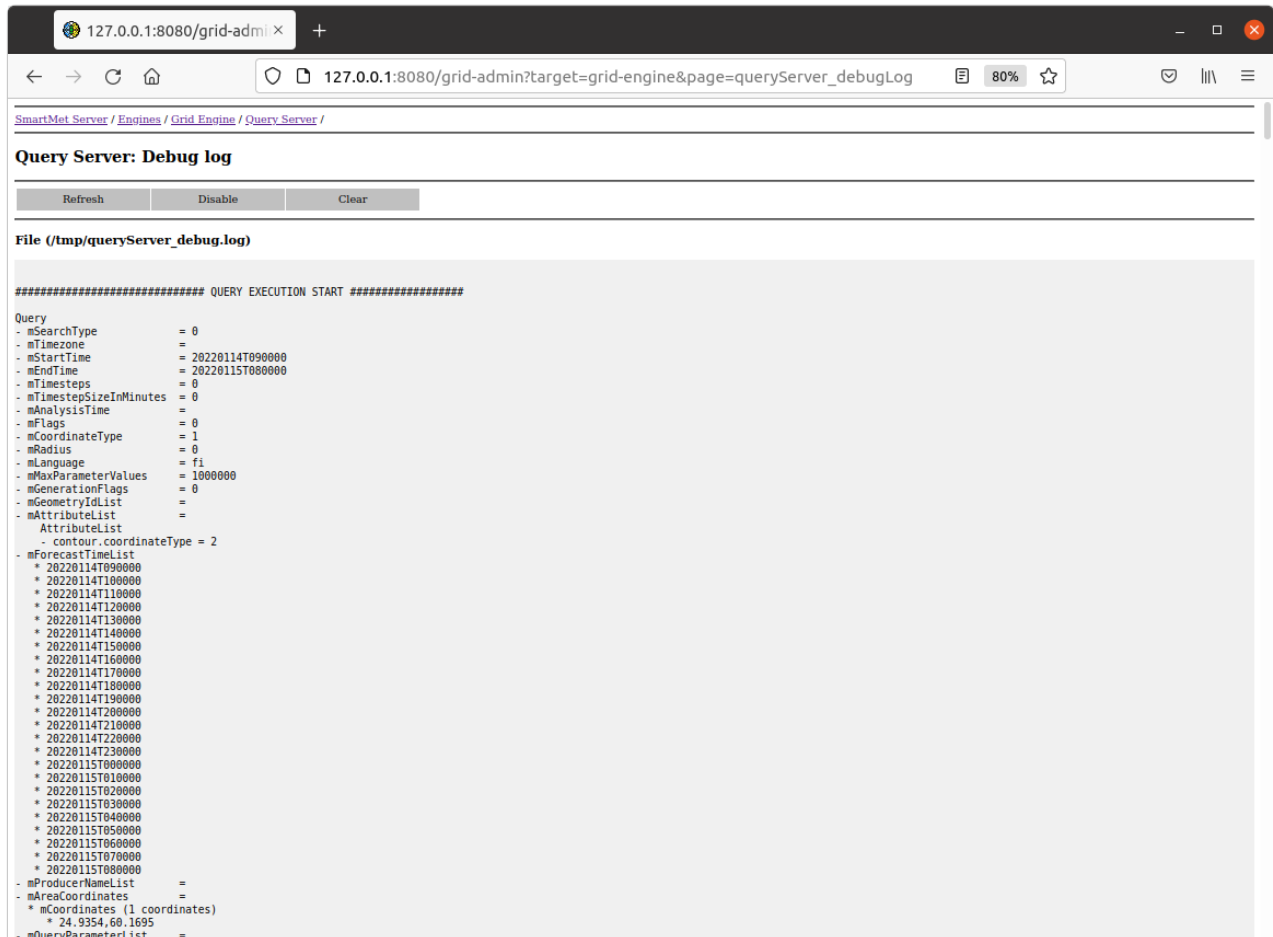
File (/home/koskelam/workspace/smartmet-engine-grid/cfg/producers.csv)

```
#####
# Producer search/selection
#####
#
# This file has two purposes:
#
# 1. It defines producers and geometries that can be searched (by Timeseries,
#    WFS, WMS, etc). If the requested producer and geometry is not in the list,
#    then the query returns nothing.
#
# 2. It defines the search order in the case where no producers are given in
#    the query. In this case, the first producer where the search location is
#    inside the current geometry, is selected.
#
# FIELDS
# 1. Producer
# 2. Geometry id
# 3. Grid size (informative, not used)
#
#####
#
SMARTMET:1096;7.6km x 7.5km;
SMARTMETMTA:1096;7.6km x 7.5km;
SMARTMETMWC:1096;7.6km x 7.5 km;
ECG:1007;11.1km x 11.1km;
ECGNTA:1007;11.1km x 11.1km;
HL2:173;7.9km x 7.2km;
HL2MTA:173;7.9km x 7.2km;
ECG:1008;8.9km x 12.9km;
ECG:1045;8.9km x 12.9km;
ECG:1090;10.1km x 11.1km;
ECG:1091;10.1km x 11.1km;
ECG:1049;17.7km x 25.8km;
ECG:1106;;;
ECG:1105;;;
ECGEPSS:1049;17.7km x 25.8km;
ECGEPSS:1068;17.7km x 25.8km;
ECGEPSS:1074;22.2km x 22.2km;
ECGEPSSMTA:1049;17.7km x 25.8km;
ECGNTA:1008;8.9km x 12.9km;
ECGNTA:1090;10.1km x 11.1km;
FRIGMTA:226;33.1km x 55.6km;
```

STEP 5:

If the previous steps did not help then we should start debugging the actual queries. You can do this by using admin-plugin.

1. Go the page “Query Server: Debug Log” and push the “Enable” button.
2. Execute a simple Timeseries query.
3. Push the “Refresh” button in the grid-admin. Now you should see something like this.



4. Check that the Query record contains all essential information related to the query.
5. Go down as long as you find line “**METHOD getGridValues()**”. The reason for the query failure usually comes very soon after this line.

```
METHOD getGridValues()
- queryType           : 0
- producers            : 1 items
  * ECG:7777
- geometryIdList       : 1 items
- producerId          : 11
- analysisTime         : 
- generationFlags      : 0
- reverseGenerations   : 0
- parameterKey         : T-K
- paramLevelId        : 0
- paramLevel           : -1
- forecastType         : -1
- forecastNumber       : -1
- queryFlags           : 0000
- parameterFlags       : 0000
- areaInterpolationMethod : -1
```

```
- timeInterpolationMethod : -1
- levelInterpolationMethod : -1
- forecastTime           : 20220115T080000
- forecastTimeT          : 1642233600
- timeMatchRequired      : 0
- locationType           : 0
- coordinateType         : 1
- areaCoordinates        : 1 vectors
- contourLowValues       : 0 values
- contourHighValues      : 0 values
- radius                 : 0.000000

- Producer and Geometry : ECG:7777
- The producer and the geometry are acceptable!
- No parameter mappings 'ECG:T-K:7777:0:-1' found!
```

In this case we tried to query a parameter from the geometry that was not used by the ECG producers and that's why the Query Server could not find any parameter mappings for this geometry.

STEP 6:

If you still have problems then you should probably use all possible logs in order to monitor the query processing. For example, you can turn on the processing logs used by the Content Server and Data Server modules. In this way you can see that there is something happening in the query processing. Unfortunately, this might be a little bit difficult for an average user.

STEP 7:

If you have “a standard” server installation, ask help from the developers. Notice that we are not necessarily able to help you if you are trying to install the system in some strange environment (container, etc.).