SmartMet Server

filesys2smartmet

Finnish Meteorological Institute

2022-01-04

TABLE OF CONTENTS

1 FILESYS2SMARTMET

1.1 Introduction

The Content Storage is a database that contains information about the available producers, generations, files and grids. The SmartMet Server uses this information in order to find correct data for the requests.

The "**filesys2smartmet**" application can be used for updating the Content Storage according to grid files found from the file system. The basic idea is that the “filesys2smartmet” application scans the file system and seeks valid grid files (GRIB1, GRIB2, NetCDF, QueryData). When it finds a grid file, it reads it and registers its content into the Content Information Storage. It can also remove this information from the Content Information Storage when files are removed from the file system.

1.2 Grid files

Registration of grid files requires that we can somehow detect the correct producer and the generation from the grid file. The easiest way to do this is to encode this information into the file name. That’s why the “filesys2smartmet” application assumes that all grid files have the name with the following format:

PROD_YYYYMMDDTHHMMSS_XXXXX.grib

In practice, the name is divided into three parts with '_' character.

The first part is a producer identifier. This identifier should be found from the producer definition file that contain all necessary information needed in order to register the current producer into the Content Storage.

The second part is a generation (forecast set) identifier that defines the generation in which the current file belongs to. The recommend format is a generation time-stamp (for example, "20180912T090000"). It identifies the generation, but at the same time it is used as the originTime / analysisTime in the Content Storage.

The last part can contain almost anything. However the file name should have a clear ending (like ".grib") so that it can be found with simple search patterns.

If a grid file name does not follow this naming structure and if it is not possible to change its name, then the "**filesys2smartmet**" program calls a LUA function that should generate the required name from the original file name. The point is that the grid file can be registered with its original name, but we need a structural name in order to find out the necessary producer and generation information. That's because we cannot relay that this information could be found from the actual grid file in such format that we could use it.

The name of the current LUA file and the name of the used LUA function is defined in the program's configuration file.

1.3 Execution

The "**filesys2smartmet**" application is usually executed in a loop, because it tries to keep the Content Storage up to data all the time. Each loop updates the Content Storage according the current available grid files in the file system. The point is that the information in the file system usually changes all the time and these changes need to be updated to the Content Storage as well.

The "**filesys2smartmet**" application is started with two parameters. The first parameter is the name of the application’s main configuration file. The second parameter is the wait time between the update loops (expressed in seconds). If the wait time is zero then the loop is executed only once.

filesys2smartmet <configurationFile> <loopWaitTimeInSeconds>

1.4 Configuration

The main configuration file of the "**filesys2smartmet**" application is a simple text file that uses JSON syntax. It can use the same “special features” that was used in the grid-engine’s main configuration file (see the “grid-engine.pdf” document). For example, it can import definitions (like passwords, connection parameters, etc.) from other files. It can also use values of environmental parameters.

The configuration file contains three main sections:

1) Initialization of the grid-files library
2) Content Source parameters (=> File system)
3) Content Storage parameters (=> Redis)

1.4.1 Grid-files library

The biggest challenge in the grid file registration is to identify grid parameters in such way that they can be used in queries. The point is that all grid file formats identify parameters differently and that’s why they need to be mapped parameters (= FMI-parameters) that are used in the SmartMet Server. There is a separate document (“grid-files.pdf”) that defines in details how this mapping is done and how it should be configured.

The grid-files library is a component that is responsible for identification of the grid parameters. We have to initialize this library before we can use it in the “**filesys2smartmet**” application. This means in practice that we have to define the name of the main configuration file for this library.

```
smartmet.library.grid-files.configFile = "$(SMARTMET_CONF)/smartmet-library-grid-files/grid-files.conf "
```

If the “**filesys2smartmet**” application does not manage to identify all grid parameters then the reason is most likely found from the configuration information of this library. More information about the mapping definitions can be found from the “grid-files.pdf” document.

1.4.2 Content Source parameters

In this case the Content Source is the file system. The Content Source can be configured like this:

```
content-source :
{
    source-id = 200
    producerDefFile = "%(DIR)/producerDef.csv"

    directories =
    [
        "/smartmet/data/ecmwf/skandinavia/pinta/querydata",
        "/smartmet/data/smartmet/grid"
    ]
    patterns =
    [
        "*.grib",
        "*.grib1",
        "*.grib2",
        "*.nc",
        "*.sqd"
    ]
    filenameFixer :
    {
        luaFilename = "%(DIR)/filenameFixer.lua"
        luaFunction = "fixFilename"
    }
}
```

The “source-id” parameter is a random number that is used in the Content Storage for identifying the source of the different data. The point is that the content information can come from multiple sources and we should be able to mark this data in such way that we can easily separate it from the other data. For example, we can easily remove data coming from source X.

The “producerDefFile” parameter is used for defining a file that contains producer definitions. The idea is that a grid file name contains just an abbreviation of the produce’s name and the rest of the producer’s information can be found from this file. The current producer file might look like this:

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

The producer file contains four fields (separated by ‘;’). The first field is the abbreviation used in the grid file names. The second field is the name that is used in query. The third field is an “official name” that can be used in documents. The fourth field is a general description of the current producer.

The “directories” parameter contains a list of directories where the grid files are searched. The search is done also in all subdirectories of the given directories.

The “patterns” parameter contains a list of file search patterns that are used during the file search. These patterns are simple “wildcard” patterns.

As was mentioned earlier, all grid files should follow a certain naming format. However, if this is not possible then we can use a LUA function in order to convert the original file name into the required format. The point is that we do not change the original file name. We just need a file name that we can use in order to find a valid producer and generation information.

The “filenameFixer” parameter is used for defining a LUA file and a LUA function that is used for fixing filenames. The LUA file might look like this.

```
function fixFilename(numOfParams,params)

    local f = params[1];  /* original filename */
    local p = params[2];  /* path */

    /* If the filename contains the “ecmwf_world_ground.grib” string, then the producer is ECG and generation timestamp
       can be picked from the filename (for example,  “202109120000_ecmwf_world_ground.grib”) */

    if (string.find(f,"ecmwf_world_ground.grib") ~= nil) then
        local new_name = "ECG_"..string.sub(f,1,8).."T"..string.sub(f,9,12).."00_.grib";
        return new_name;
    end

    return f;   /* Returning the original name */

end


/* This function should be in all LUA files. It is used for registering functions that can be called */

function getFunctionNames(type)

    local functionNames = '';
    if (type == 6) then
        functionNames = 'fixFilename';
    end
    return functionNames;

end
```

1.4.3 Content Storage parameters

The "**filesys2smartmet**" application can connect to the Content Storage in different ways. When the Content Storage is a Redis database then the simplest way to use Redis is via its TCP connection. If the Content Storage is a CORBA-server then we just need to define its IOR (International Object Reference). The Content Storage can be also a HTTP-server, which means that we need to define its URL.

The configuration of the Content Storage connection is quite easy (especially if the current Content Storage is already up and running). First, we need to define the type of the Content Storage and after that we just fill the configuration parameters related to the current type.

For example, if we select the type to be "redis" then we just fill the rest of the "redis" related parameters and ignore parameters related to other types (corba, http, file).

```
content-storage :
{
    # Content storage type (redis/corba/http)
    type = "redis"

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


    corba :
    {
        ior         = "$(CORBA_CONTENT_SERVER_IOR)"
    }

    http :
    {
        url         = "$(HTTP_CONTENT_SERVER_URL)"
    }
}
```

The same Radon database can be used by multiple SmartMet Server installations. The "tablePrefix" parameter is used in order to separate different installations. It is a kind of "namespace" definition that is added in the beginning of table names. For example, the SmartMet Sever uses table names like "producers", "generations", "files", "content", etc. Because the Radon database does not contain namespace definitions itself, we just add the "table prefix" on the front of these table names (=> "a.producers", "a.generations", etc.).

1.4.4 Logs

The "**filesys2smartmet**" application has two different logs: 1) the processing log and 2) the debug log

The processing log is a formal log that shows starting times of different processing phases. The processing log can be enabled in the main configuration file.

```
processing-log :
{
    enabled      = true
    file         = "/tmp/filesys2smartmet_processing.log"
    maxSize      = 100000000
    truncateSize = 20000000
}
```

The "maxSize" parameter is the maximum size limit for the log file. After that it is automatically truncated, which means that the old data in the beginning of the file is automatically removed. The "truncateSize" is the preferred size of the log file after the truncate operation.

The debug log is used for debugging purposes. The debug log can contain whatever debug information that the developer has wanted to print into it (parameter values, code phase indicators, warnings, etc.). It is configured in the same way as the processing log.

```
debug-log :
{
    enabled         = false
    file            = "/tmp/filesys2smartmet_debug.log"
    maxSize         = 100000000
    truncateSize    = 20000000
}
```