SmartMet Server

radon2smartmet

Finnish Meteorological Institute

2022-01-03

TABLE OF CONTENTS

1 RADON2SMARTMET

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.

Finnish Meteorological Institute uses the "**radon2smartmet**" application in order to keep the Content Storage up to date. This application just updates the Content Storage according to information that it gets from the other FMI's internal database (Radon). Unfortunately, this application is useless for the users who do not have the Radon database.

1.2 Execution

The "**radon2smartmet**" 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 Radon information. The point is that the information in Radon changes all the time and these changes need to be updated to the Content Storage as well.

The "**radon2smartmet**" 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.

radon2smartmet <configurationFile> <loopWaitTimeInSeconds>

1.3 Configuration

The main configuration file of the "**radon2smartmet**" 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 two main sections:

1) Content Source parameters (=> Radon)
2) Content Storage parameters (=> Redis)

1.3.1 Content Source parameters

In this case the Content Source is the Radon database, which is a PostgreSql database. The Content Source can be configured like this:

```
content-source :
{
    source-id = 100
    producerFile = "%(DIR)/producers.cfg"

    radon :
    {
        connection-string = "host=radon.fmi.fi dbname=radon user=uuuuuu password=xxxxxxxxxxxxxxxx"
    }
}
```

The "connection-string" parameter is need in order to define connection to Radon database, which is a PostgreSql database.

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 “producerFile” parameter is used for defining a file that contains a list of producers that we want to update in the Content Storage. The point is that the Radon database might contain hundreds of different producers and we do not necessary need them all. The current producer file might look like this:

```
ECG,ECGMTA;1;2;1
HL2;0;3;2
SMARTMET,SMARTMETMTA;1;1;1
ECGERA5;10;100
ECMOS2;0;3;5
```

The producer file contains four fields (separated by ‘;’).

The first field contains one or multiple producer names (separated by ‘,’). Multiple producer names are written on the same line when there is a need to synchronize the publishing of the generations belonging to these producers. The idea is that generations can be published only if all synchronized producers have the same generation available. This might be necessary if the generations have remarkable relationships or dependencies from each other. For example, ECGMTA generations are usually calculated from the ECG generations, which means that these generations are related to each other.

The second field is used for activating this synchronization (1 = enabled, 0 = disabled).

The third field indicates the update starting order (= loop index number) of the current producer(s). The point is that there might be a need to update some producers immediately when the “radon2smartmet” application is started. On the other hand, some producers can wait this update a little bit longer.

The fourth field indicates the update interval of the current producer expressed in loop numbers. Value 1 means that the producer must be update every time when the loop is executed, value 2 means that the producer must be updated every second time when the loop is executed, and so on.

1.3.2 Content Storage parameters

The "**radon2smartmet**" 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.).

1.3.3 Logs

The “**radon2smartmet**” 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 looks like this:

```
[2021-11-30/10:23:54][160][PRODUCER-ADD;OK;SMARTMET]
[2021-11-30/10:23:54][161][GENERATION-ADD;OK;SMARTMET:20211129T000000]
[2021-11-30/10:23:55][162][GENERATION-ADD;OK;SMARTMET:20211130T000000]
```

The processing log can be enabled in the main configuration file.

```
processing-log :
{
    enabled         = true
    file            = "/tmp/radon2smartmet_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/radon2smartmet_debug.log"
    maxSize         = 100000000
    truncateSize    = 20000000
}
```