|  |  |  |  |
| --- | --- | --- | --- |
| logofire-standardresolution | jaune | logo_ce-en-rvb-hr | 7plogo |

|  |  |
| --- | --- |
| Project Acronym | **Fed4FIRE** |
| Project Title | **Federation for FIRE** |
| Instrument | **Large scale integrating project (IP)** |
| Call identifier | **FP7-ICT-2011-8** |
| Project number | **318389** |
| Project website | **www.fed4fire.eu** |
| Experiment | **GEO-Cloud** |

**Satellite System Development**

|  |  |
| --- | --- |
| Work package | WP10 |
| Task | T10.1.2 GEO-Cloud Experiment |
| Due date | 10/01/2014 |
| Submission date | 10/01/2014 |
| Report Lead |  |
| Version | 1.0 |
| Authors |  |
| Reviewers |  |

|  |  |
| --- | --- |
| **Document ID** | **GEO-Cloud-** |
| Abstract | This document provides technical information about the GEO-Cloud development of the in orbit system. |
| Keywords | Deliverable |

|  |  |  |  |
| --- | --- | --- | --- |
| Nature of the document | R | Report | X |
| P | Prototype |  |
| D | Demonstrator |  |
| O | Other |  |
| Dissemination level | PU | Public |  |
| PP | Restricted to other program participants (including the Commission) |  |
| RE | Restricted to a group specified by the consortium (including the Commission) |  |
| CO | Confidential, only for members of the consortium (including the Commission) | X |

**Disclaimer**

*The information, documentation and figures available in this deliverable, is written by the Fed4FIRE* (Federation for FIRE) *– project consortium under EC co-financing contract FP7-ICT-318389 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.*

**Executive Summary**

In this document…

**Acronyms and Abbreviations**

|  |  |
| --- | --- |
| AOI | Area of Interest |
| API | Application Programming Interface |
| BF | BonFIRE |
| CSV | Comma Separated Values |
| CSW | Catalogue Services for the Web |
| DBMS | Data Base Management System |
| EO | Earth Observation |
| FTP | File Transfer Protocol |
| GIS | Geographic Information System |
| GMES | Global Monitoring for Environment and Security |
| GPS | Global Positioning System |
| GPU | Graphics Processing Unit |
| GQ | Golden Quadrilateral |
| GS | Ground Station |
| GSD | Ground Sample Distance |
| HMA | Heterogeneous Missions Accessibility |
| HPC | High Performance Computing |
| HTTP | Hypertext Transfer Protocol |
| INSPIRE | Infrastructure for Spatial Information in Europe |
| ISO | International Organization for Standardization |
| KML | Keyhole Markup Language |
| L0 | Level 0 processor |
| L1A | Level 1 A processor |
| L1B | Level 1 B processor |
| L1C | Level 1 C processor |
| LEO | Low Earth Orbit |
| LTAN | Local Time of Ascending Node |
| NASA | National Aeronautics and Space Administration |
| NHAI | National Highways Authority of India |
| OGC | Open Geospatial Consortium |
| PNOT | Plan Nacional de Observación del Territorio (Spanish Plan of Territory Observation and Urbanism) |
| PNT | Plan Nacional de Teledetección (Spanish Plan of Remote Sensing) |
| PP | Product Processor |
| RDBMS | Relational Database Management System |
| REST | Representational State Transfer |
| Sat | Satellite |
| SCO | Snow Cover Area |
| SDI | Spatial Data Infrastructure |
| SFTP | Secure File Transfer Protocol |
| SOA | Service Oriented Architecture |
| SQL | Structured Query Language |
| SSO | Sun-Synchronous Orbit |
| SWE | Snow Water Equivalent |
| TBD | To be defined |
| TMS | Tile Map Service |
| UTGC |  |
| VM | Virtual Machine |
| VW | Virtual Wall |
| WCS | Web Coverage Service |
| WFS | Web Feature Service |
| WMS | Web Map Service |
| WMS-C | Web Map Tile Caching |
| WMTS | Web Map Tile Service |
| WPS | Web Processing Service |
| WS | Web Service |
| XML | eXtensible Markup Language |

**Table of Contents**

[1 Introduction 8](#_Toc382217683)

[2 Image Acquisition 9](#_Toc382217684)

[3 Image downloading 10](#_Toc382217685)

[4 Bibliography 11](#_Toc382217686)

# Introduction

In the satellite system implementation the data required to simulate the acquisition and downloading of the images has to be provided. Some located scenarios have been designed in order to reduce the amount of data to work with during the emulation. This way, the simulation is shortened to the time needed to acquire and download certain areas of interest (AOI).

In this document, the main aspects that appear during the simulation are explained in order to make understandable why some data are required and expose the different situations when acquiring and downloading the scenarios.

# Image Acquisition

The first step to be implemented in the system is the acquisition of the six predefined scenarios (See GEO-Cloud-D10.8-Detailed design report-2014-01-31) with the satellite constellation. The area of interest (AOI) in each scenario can be acquired by one or more “main satellites” depending of the size of the AOI relatively to the scene size (in this system, a scene covers 160km x 160km), and during this acquisition the images acquired by the rest of satellites (which are not imaging AOI) will be used as noise to analyse the performance of the system in a case as realistic as possible. It has to be noticed that the main satellites can also acquire some images different from the AOI along the duration of each scenario that will also be used as noise in the system. The time is referenced to a 0 instant in which the simulation of the system starts in the software employed for this analysis.

Depending on the relative sizes of AOI and scene, three different situations are presented when a satellite is taking images:

1. Simple acquisition: All the AOI (at least the part to be imaged by this satellite) fits in just one scene.



Figure 1. Simple acquisition.

1. Multiple consecutive acquisitions: All the AOI to be acquired by a satellite fits in a strip with some scenes in it.

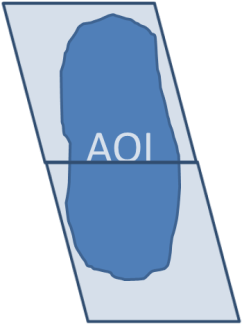


Figure 2. Multiple consecutive acquisitions.

1. Multiple non-consecutive acquisitions: All the AOI to be acquired by a satellite fits in a strip but there are some scenes between acquisitions that have to be acquired in order to complete the world map but it is not necessary for the scenario.

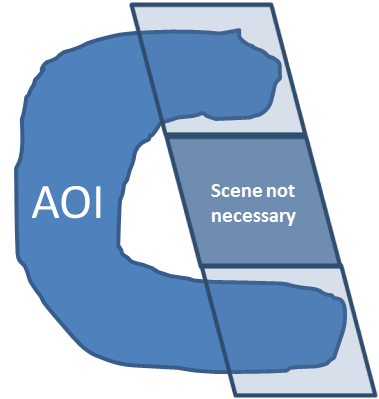


Figure 3. Multiple non-consecutive acquisitions.

For each scenario, the following parameters are required:

* Start time: The scenario starts when the first main satellite begins the acquisition of the first portion of AOI.
* End time: It is the time when the last main satellite (the last satellite that acquire some portion of the AOI, if the AOI fits in just one image then the first and last main satellites are the same) finishes the downloading of the AOI.
* Main satellites: Numbered from 1 to 17, the main satellites responsible of the acquisition of some portions of the AOI for each scenario shall be detailed.
* AOI acquisition time: It is the time during each main satellite is imaging the AOI. It is a time list with the start and end for each main satellite. This time is a multiple of 23.4 seconds, which is the needed time to acquire one scene of 160km x 160km (the satellite is overflying the Earth surface at 6.84km/s, then 23.4s is required to acquire 160km along track).
* AOI acquired images for each main satellite: During the AOI time acquisition, each main satellite can acquire one or more scenes including the AOI depending on the size of it. The number of scenes times 23.4 seconds is the AOI acquisition time.

As an example, the following data has been obtained from “Scenario 2: Infrastructure monitoring. Affection in railway infrastructures by sand movement in desert areas”:

Table 1. Example of data of image acquisition for Scenario 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario | Start time | End time | Main Satellites | Start Acquisition | End Acquisition | AOI Images |
| 2 | 54060 | 54753.4 | 4 | 54060 | 54083.4 | 1 |
|  |  |  | 3 | 54390 | 54413.4 | 1 |
|  |  |  | 2 | 54730 | 54753.4 | 1 |

# Image downloading

After the acquisition, images shall be downloaded to ground stations through the antenna network designed in the system. All the scenarios finish when the last main satellite downloads completely the last portion of AOI. During the simulation, the rest of satellites not overflying the AOI would be imaging other places and these images will be also downloaded and processed in parallel to the AOI images, in order to simulate a case as realistic as possible.

Because of the limitations in the test beds (storage, etc.) it is necessary to scale the data involved in these simulations. It has been decided to change the designed compression rate from lossless compression 2:1 to a loss 14:1. With this change, the data volume has been reduced and the size of every 160kmx160km scene after compression is 288MB.

For the downloading step, the satellite has to be inside the visibility cone of a ground station. During these accesses, the satellites download images at a rate of 160Mbps. Part of these images are in memory before entering the visibility cone and others are simultaneously acquired during the downloading (as in the case of AOI images, which are downloaded at the same time of the acquisition if the satellite is inside a visibility cone during the acquisition). This is why it is also required to provide the time of beginning and ending of the accesses of all the satellites of the constellation during each scenario. The duration of the accesses could not be multiple of the time to download an image, in case an image is not downloaded completely it would be suppose this image is completely downloaded in the next access. According to the data rate of 160Mbps and the specifications of the payload (channel bands, digitalization, resolution, ancillary data, compression rate…) one image is downloaded every 14.4s. For the memory management, a LIFO (last in, first out) procedure has been decided, this decision implies that the latest acquired images will be downloaded as soon as possible, and because of that when a satellite is inside the visibility cone of a ground station the images being acquiring will be simultaneously downloaded. Note that because of differences between the rate of acquisition and downloading, and also because of the compression process, the downloading of images that are being acquiring do not occupy all the bandwidth of 160Mbps, the portion not used is employed to download on board storage also with a LIFO procedure.

As an example, the data of the accesses for the Scenario 2 are including in the following table:

Table 2. Example of data of accesses for Scenario 2.

| Sat | Ground Station | Start time | End time | Duration (sec) |
| --- | --- | --- | --- | --- |
| 1 | Krugersdorp | 54184.2 | 54516 | 331.8 |
| 2 | Dubai | 54549.4 | 54753.4 | 204 |
| 2 | Krugersdorp | 54060 | 54127.3 | 67.3 |
| 3 | Dubai | 54184 | 54662.1 | 478.1 |
| 4 | Dubai | 54060 | 54337.5 | 277.5 |
| 4 | Svalbard | 54660 | 54753.4 | 93.4 |
| 5 | Svalbard | 54316.3 | 54753.4 | 437.1 |
| 6 | Prince Albert | 54708.1 | 54753.4 | 45.3 |
| 6 | Svalbard | 54060 | 54639.4 | 579.4 |
| 7 | Prince Albert | 54362 | 54753.4 | 391.4 |
| 7 | Svalbard | 54060 | 54296.7 | 236.7 |
| 8 | Prince Albert | 54060 | 54577.9 | 517.9 |
| 9 | Prince Albert | 54060 | 54246.8 | 186.8 |
| 15 | Troll | 54419 | 54670.3 | 251.3 |
| 16 | Troll | 54085.6 | 54303.4 | 217.8 |
| 17 | Krugersdorp | 54495.4 | 54753.4 | 258 |

# Getting the satellite data

To simulate the behaviour of satellites, the real data of each scenario is necessary.

First point of this is that the constellation of satellites has been simulated in a program. This software is named Satellite Tool Kit (STK). The data of each satellite has been introduced like such as its own orbital data. After that, the data of scenarios can be getting easily exporting all of them to a “CSV” format file. The format of the resulting file, for example a slice of the Lorca scenario, looks at this in Figure 4:

|  |
| --- |
| "GEO-Cloud\_005-To-Troll - Access","Start Time (EpSec)","Start Time (UTCG)","Stop Time (EpSec)","Stop Time (UTCG)","Duration (sec)"  1,63638.000,11 Aug 2014 10:10:38.000,63661.400,11 Aug 2014 10:11:01.400,23.400  "GEO-Cloud\_006-To-Troll - Access","Start Time (EpSec)","Start Time (UTCG)","Stop Time (EpSec)","Stop Time (UTCG)","Duration (sec)"  7,63638.000,11 Aug 2014 10:10:38.000,63661.400,11 Aug 2014 10:11:01 |

Figure 4 Piece of file information

As can be seen in figure above, each access to a ground station area realized by Satellite is written. This action is headed by the sentence “GEO-Cloud\_XXX-To-YY”, being XXX the satellite has realized the action and YY the ground station in whose area the satellite has been. The main part of header has seen before, because the rest of the line indicates the mean of the columns. Start Time (EpSec) indicates the relative time respect to the moment of the orbital simulation starts running; Start Time (UTCG) indicates the hour in format UTCG respect the orbital simulation starts running; the Stop Time (EpSec) and Stop Time (UTCG) shows the times when the satellite leaves from the area of ground station and follow the same format that Start times seen before. Also, the duration is shown, that is the time difference between the same format times.

Furthermore, the name of this file is special. The format of the file name must be like this pattern:

“Scenario\_<NUM>\_<SCENE>.csv” where NUM indicates the number of scenario and Scene is the name of this scenario, but the most important of this format are the “\_” symbol between Scenario and number and between number and scene, and the file must finish in “.csv” format.

In addition to above, it´s necessary to know when the main satellites catches AOI. For this, a file that contains all times in which satellites catches AOI, order by the scenario in which the event occurs, has been built. The format of this file is depicted in Figure 5.

|  |
| --- |
| Scenario,Start Sce,End Sce,Sat,Start sat,End Sat,Images  1,63638,63661.4,11,63638,63661.4,1  ,,,,,,  2,54060,54753.4,4,54060,54083.4,1  ,,,3,54390,54413.4,1  ,,,2,54730,54753.4,1  ,,,,,,  3,62480,63865.4,15,62480,62503.4,1  ,,,14,62821,62844.4,1  ,,,13,63161,63184.4,1  ,,,12,63501,63524.4,1  ,,,11,63842,63865.4,1 |

Figure 5 AOI Data of all scenarios

The building of this file has been made manually, so the format doesn´t follow any external rules but also is in “CSV” format. The proposal format follows the simplicity principle (only putting the necessary info). The useful fields for describing the behaviour of satellites in the AOI are the next:

|  |  |
| --- | --- |
| **Column Title** | **Function** |
| Scenario | Shows in which scenario the satellite will behave this way. |
| Star Sce | Indicates when the scenario starts. It´s no necessary because this info appears in the file above but in this file, simplified the reading. |
| End Sce | Indicates when the scenario finishes. The same explanation as above. |
| Star Sat | The time when the satellite starts to catch AOI. |
| Sat | Indicates the number of main satellite |
| End Sat | The time when the satellite leaves from AOI. |
| Images | This number of useful images that can be catch by the satellite. |

As it can be looked, the files need to be pre-processed. In the section below will be explained how the reading process is performed and the information is storage in a database.

## Processing files

In this subsection, the database to store the data will be depicted and its model architecture will be explained. Also, the program that gets the data from files and stores in the data base will be described.

### Data base

In order to storage the information that contains the files shown in Figure 4 and Figure 5, is necessary a data base. The selected data base management system (DBMS) is MySQL. It has been selected by to be friendly use and familiar, among other causes.

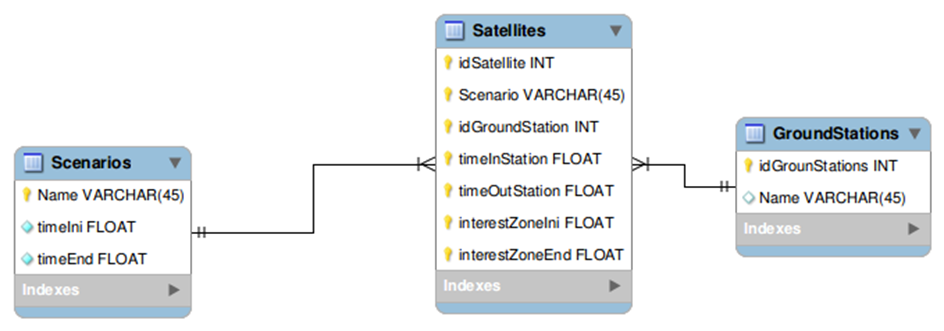
The design of data base architecture has been the shown in the Figure 6.

Figure 6 Data Base Architecture

The architecture is easily understandable. There are three tables: Scenarios, Satellites and GroundStations. Now these tables will be described:

|  |  |  |
| --- | --- | --- |
| **Table** | **Function** | **Relationship** |
| Scenarios | Contains the name of the scenario, the start time and the end time. Its primary key\* is the Name column. | None |
| GroundStations | Contains the id of Ground Station and how it is named. Its primary key is the idGroundStation. | None |
| Satellites | Contains the idSatellite, the scenario and the ground station the event occurs, and the necessary times that they are the time the satellite enters in the visibility cone and it leaves of the visibility cone, the time when the satellite enters in AIO and when it leaves of AIO. | This table relates with Scenarios and GroundStations table. This is because Scenario and idGroundStation are foreign keys\*\*. The characteristics of this field are On delete cascade and On update cascade. This means when an object contained in a table such GroundStation or Scenarios is deleted, also is deleted the object that references it automatically. |

Note \*: The primary key only and exclusively identifies an object, for example a scenario, a ground station or a satellite.

Note \*\*: The foreign key binds two tables, by linking across a field. For example in this case, the column from Satellite names Scenario references an object that exists in the Scenarios table, so the Satellite.Scenario points to Scenarios.Name.

### File processing

The procedure for processing the files will be explained in this section.

The program developed for this proposal names “setDatabase.py”. It has developed in Python 2.7 but is also compatible with all Python versions.

For executing this program it is necessary to proportionate some arguments that are explained here:

|  |  |
| --- | --- |
| **Argument Position** | **Meaning** |
| 1 | IP direction of host that contains the MySQL data base |
| 2 | The relative path or absolute path of the file that contains the information of all scenarios likes Figure 5 |
| 3… | The files that contain the events occurred in a particular scenario. For example can be introduce some files describing scenarios as Figure 4 do it. |

An example execution could be like this:

* python setDatabase.py 192.168.0.2 All\_scenarios\_file.csv Scenario\_1\_Example1.csv Scenario\_2\_Example2.csv Scenario\_3\_Example3.csv

Now, the workflow of the program will be described step to step:

1. The all data into data base is cleaned to avoid inconsistencies.
2. The GroundStation table is filled with the ground stations information (name and id).
3. The Scenario table is filled inserting all depicted data above.
4. Finally, the Satellite table is filled. This process is the most complex why it is necessary to realize a join of the all scenarios file and the present processed file of a concrete scenario. This means that for each scenario file, it must to localize witch satellite corner is being processed, to look for if there are any AOI for this satellite and scenario and if there was, to select the AOI for this explored satellite corner.

After that the data base is filled and usable. Surely, while the above process is made, the program have to connect to the data base server to put into the respectively information.

# Satellite Simulation

In this section, the behaviour of satellites and how it is implemented will be described. Also will be discussed the different approaches for its implementation and how finally, will be implemented. Finally, the execution of the satellite will be showed.

## Satellite Behaviour

The satellite behaviour is very easy. The functionality can be resumed in some parts:

* If the satellite is not in a visibility cone, two cases can take place:
  + The satellite is not in an interesting area, so it does nothing.
  + The satellite is in AOI, so the satellite must acquire useful images.
* If the satellite is in a visibility cone, next cases can occur (described in Section 3):
  + The satellite is not in an interesting area, so the satellite only will be downloading to the ground station useless images while it is in this area.
  + The satellite is in an interesting area and this finishes before the satellite goes out of the visibility cone, it will be downloading to the ground station as Section 3 describes, multiplexing the channel.
  + The satellite is in an interesting area but the visibility cone finishes before the satellite goes out of the interesting area.

## Detailed design and Implementation

In this section the low-level design and implementation will be explained. Also, the satellite simulation software workflow will be depicted.

The satellite behaviour has been explained above. For the Geo-Cloud experiment, it is necessary to simulate the satellite constellation, so in the next sections the design and implementation will be explained for doing an exactly simulation of the complete experiment.

### Detailed Design

In this section, the detailed design will be explained. Also, all dependencies for simulating all the satellites in software will be shows.

The satellite behaviour software has been developed once. This software will able to execute a parameterized satellite so it implies that each satellite will be execute the same software but with different inputs. These parameters indicate which satellite the software will simulate and also in which scenario the simulation will be done.

In addition, the satellite software needs that all the data corresponding of simulated scenario is into the data base. For this, the data base software has to be executed how the Section 4.1.1 describes.

Once, all necessary data is in the data base, the satellite software can be executed.

The satellite software simulates one selected main satellite of the constellation in a scenario, so it needs the inputs as follows:

|  |  |
| --- | --- |
| **Argument** | **Meaning** |
| Identity | Identification of satellite |
| Scenario | Indicates the simulated scenario |
| Host | The IP direction of host in which the data base is located |

When the inputs have been reached, the software starts the execution of the satellite behaviour. The steps that the software must realise are depicted as follows:

* First, the satellite identifies who is itself using the identity input.
* Then, the satellite gets all own data of this scenario from data base using the host to connect with.
* At this point, the satellite knows how will be its interactions with the scenario and the ground stations, so it can schedule the interactions with the ground stations along the time. The possible interactions are described in Section 5.1.
* When all the interactions have been scheduled, the satellite can start to execute the simulation of scenario. This execution will be made sequentially, sorted by the time in which the action in the scenario is taken.
* When the last scheduled task has been executed, the scenario simulation finishes.
* Finally, conclusions can be reached as results of the execution (for example, sent images to ground stations).

The Figure 7 shows how the above process is realized and shows the interaction with other subsystems as data base system:

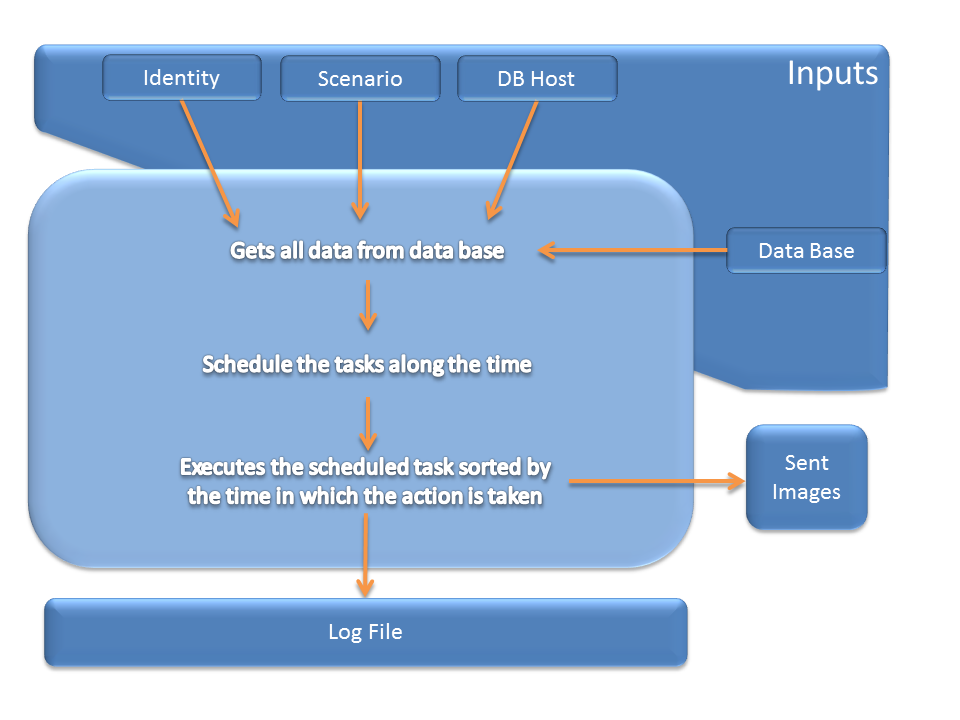


Figure 7 Satellite Execution

### Implementation

In this section, the low-level implementation of the satellite simulation software will be explained. The libraries used for to develop and how the simulation time is treated will be commented. Also, the workflow of the simulation software will be depicted.

The implementation has been realized in Python2.7, using the libraries above.

The python´s libraries necessaries to be able to implement the software will be listed as follows:

|  |  |
| --- | --- |
| **Python Library** | **Function** |
| Sys | System library |
| OS | Operative system interactions supplies |
| MySQLdb | Supports the interactions between the software and the data base connection for querying the data |
| Sched | Library that permits to realise the scheduling of the task respect of the time |
| Time | For managing the time |
| Socket | Library for creating and establishing connections with other host |
| Pdb | Used for debugging the software |

About the used times, there are two types of times with the software works. The firsts are the times that the STK software gives. Those times are in relative format of STK, that is to say that are the relative times on the beginning the scenario simulation in STK. It causes that the total simulation real time was unreachable because will cause that the execution of the experiment last a long time.

To deal this problem, the time with the software will work needs to be relative. The proposed solution consists on to take the difference between the final time and the start time of an action realised by the satellite. Then, this result is divided by 10, obtaining the duration of the real interval whose value is more able to be simulated but ten times lower.

Now, the detailed implementation will be explained. This explanation will be made as a consecutive list of actions that are implemented in the software. The software workflow is listed as follows:

1. The software gets the inputs (host in which data base is located, the identity of satellite to simulate and the scenario to simulate).
2. It creates a connection with the data base using the MySQLdb library.
3. Querying to data base, the software gets the interesting data for the satellite in the simulated scenario. Also, the scenario data as start time and end time is applied for too. There is an important thing to highlight. This is that the query made to the data base contains the particle “order by timeInZone”. This means that the data base will return the result of the query order by, in increasing order, the actions done by the satellite. Furthermore, other query is made for getting the IP directions of every ground stations.
4. When all data of step above is caught, the scheduling process starts. This process consists on, over all actions done by the satellite, to schedule actions whose may be the next:
   * If the current zone hasn´t got an interesting area, the area is scheduled starting in the visibility zone start time, realizing the “NotInterestingZone” function.
   * If the current zone has got an interesting area, the next cases may be:
     + If the interesting zone start time is smaller than visibility zone start time, the acquired data before the visibility cone start time has to be calculated. This is made calculating the difference between those times and multiplying by the acquisition images rate and all this divided by the compression rate. Then, a task will be scheduled. This will realize the “InterestingZone” function during the interesting area time and the acquired data is passed for sending to the ground station.
     + If the interesting zone start time is the same as visibility zone start time, a task will be scheduled. The task will realize the “InterestingZone” function during the interesting area time.
     + If the interesting zone starts after that the visibility cone begins, two tasks will be scheduled. The first task will realize the “NotInterestingZone” function between the time in which the visibility cone starts and the interesting zone begins. The second task will realize the “InterestingZone” function during the time that the interesting zone is.
     + In all cases, if the time of interesting area ends is less than the time when the visibility cone ends, a new task is scheduled realizing the “NotInterestingZone” function because the satellite catch useless images.
     + In all cases but not in the last, the task that realizes the “OutOfVisibility” function is scheduled.

The “NotInterestingZone” function, during the time of this zone is being, useless images are caught and sent to the ground stations.

The “InterestingZone” function, during the time of this zone takes place, sends useful images to the ground stations.

The “OutOrVisibility” function all the time that it is taking place, do nothing, only passed the time.

In order to understand the process explained above, a flow diagram is showed in Figure 8:

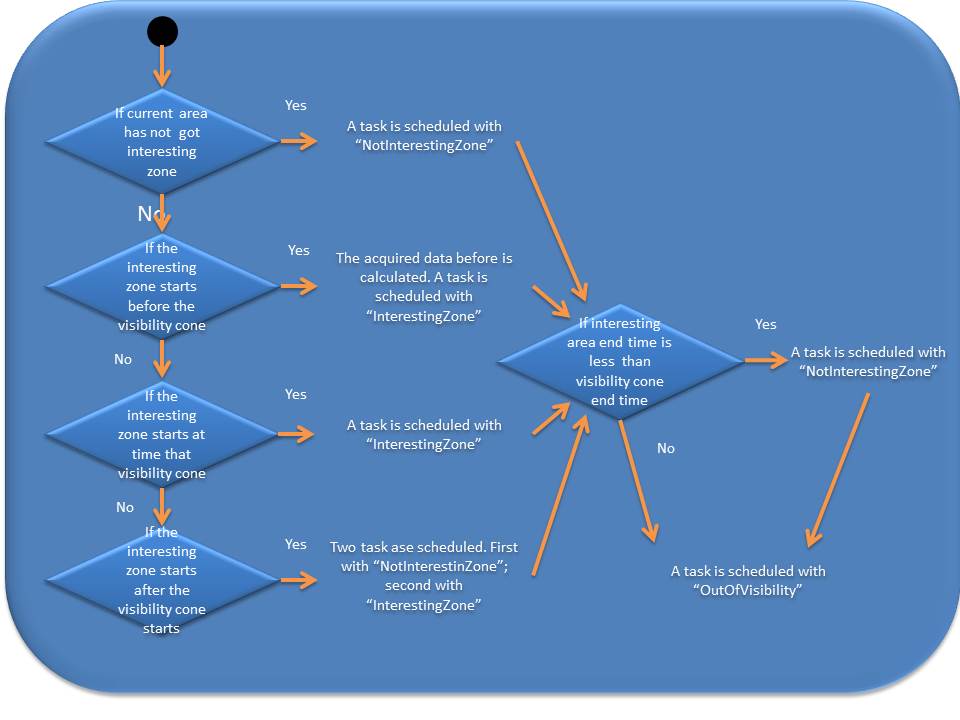


Figure 8 Schedule Process

### Execution

The satellite software is developed in a multiplatform language, but it is restricted to be executed in UNIX operative systems because there are many dependencies with some packets and the file system.

The execution of the satellite software it is realising as next:

> python satellite.py <IDSAT> <SCENARIO> <DBHOST>

Where:

* IDSAT is the satellite identity. This value must be an integer.
* SCENARIO is the scenario to simulate. Must be an integer.
* DBHOST is the host where the data base is located. Must be a hostname or an IP direction.

# Ground Station Simulation

In this section, the ground station behaviour will be explained. Also, the software that implements this behaviour will be low-level commented.

## Ground Station Behaviour

In this section, the ground station behaviour will be explained. The functions of ground stations are:

* To receive the images sent by the satellites.
* Create the files which the Orchestrator will be downloaded for processing and publishing in the Archive and Catalogue subsystem.

The ground station behaviour can be resumed in some parts:

* If in the ground station a new connection is opened, a new process is created for receiving the images sent by the satellite.
* When the satellite has sent all images, the images are created in the ground station storage and the process is closed.
* The parts above are over a loop which is listening in a port for inputs connections.

### Detailed Design

In this section, the detailed design will be explained. Also, all dependencies for simulating all the ground stations in software will be shows.

The ground station behaviour software has been developed once. This software will able to execute a parameterized ground station, so each ground station will be execute the same software but with different inputs. These parameters does the software will simulate the ground station of the scenario and also, the scenario in which this simulation will be done.

In addition, the ground station software needs that all the data corresponding of simulated scenario is into the data base. For this, the data base software has to be executed how the Section 4.1.1 describes.

Once, all necessary data is in the data base, the ground station software can be executed.

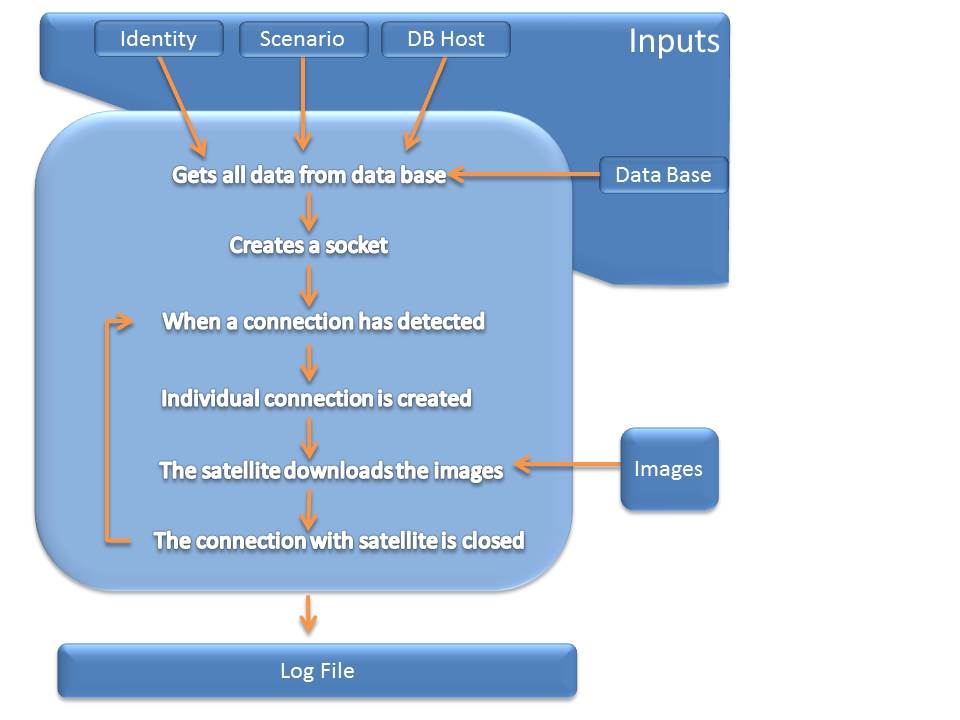
The ground station software simulates one ground station in the scenario, so it needs the inputs as follows:

|  |  |
| --- | --- |
| **Argument** | **Meaning** |
| Identity | Identification of ground station |
| Scenario | Indicates the simulated scenario |
| Host | The IP direction of host in which the data base is located |

When the inputs have been reached, the software starts the execution of ground station behaviour. The steps that the software must realise are depicted as follows:

* First, the ground station identifies who is itself using the identity input.
* Then, the ground station gets all own data of this scenario from data base using the host to connect with.
* At this point, the ground station knows who is and in which scenario the simulation is taken place. The possible interactions are described in Section 6.1.
* Then, the ground station creates a socket, in which the ground stations will be able to connect and to transfer the images.
* This created socket is always listening in connections.
* If there is any input connection, the ground station serves this petition creating an individual connection for this supplicant satellite.
* In that connection, the satellite can download the all images it storages.
* When the satellite finishes downloading the images, the connection is closed.

The Figure 7 shows how the above process is realized and shows the interaction with other subsystems as data base system:



It´s clear that the download rate of the real simulation will not be able to be represented. That is why the satellite has 160 Kbits per second of download rate. If the simulated seconds corresponds with ten seconds in a real simulation, the download rate is unreachable. To front this issue, the selected approach consists on not to send all the data required. In other words, each second the satellite software sends a packet that contains the type of information (interesting zone or not) to the ground station. When the ground station receives it, the station knows which type of packet is and depending on this, adds the packet information to interesting received data or not interesting received data.

### Implementation

In this section, the low-level implementation of the ground station simulation software will be explained. The libraries used for to develop and how the simulation time is treated will be commented. Also, the workflow of the simulation software will be depicted.

The implementation has been realized in Python2.7, using the libraries above.

The python´s libraries necessaries to be able to implement the software will be listed as follows:

|  |  |
| --- | --- |
| **Python Library** | **Function** |
| Sys | System library |
| OS | Operative system interactions supplies |
| MySQLdb | Supports the interactions between the software and the data base connection for querying the data |
| Select | Library that permits querying to operative system how many connections there are and serves its. |
| Time,Datetime | For managing the time |
| Socket | Library for creating and establishing connections with other host |
| Pdb | Used for debugging the software |
| Signal | For managing the new processes |
|  |  |

Now, the detailed implementation will be explained. This explanation will be made as a consecutive list of actions that are implemented in the software. The software workflow is listed as follows:

1. The software obtains the inputs (host in which data base is located, the identity of satellite to simulate and the scenario to simulate).
2. It creates a connection with the data base using the MySQLdb library.
3. The IP direction of the host where the ground station is executing is obtained and the data base is updated with the direction of the ground station.
4. Querying to data base, the software gets the interesting data for the ground station in the simulated scenario.
5. A socket is created, and listening the input connections.
6. Starts a loop that contains a select operation. This operation gets the connections has been realized with the ground station socket.
7. For each detected connection, a new process is created. This process will receive all data that satellite will send. When the satellite finishes sending images, the process creates the downloaded images and it is killed.

### Execution

The ground station software is developed in a multiplatform language, but it is restricted to be executed in UNIX operative systems because there are many dependencies with some packets and the file system.

Also, the software needs to know the IP direction assigned to its ETH0 interface. If this interface is not connected, the software looks for the WLAN0 interface. This information is accomplished from the UNIX file “/etc/hosts”.

The execution of the satellite software it is realising as next:

> python groundstation.py <IDSAT> <SCENARIO> <DBHOST>

Where:

* IDSAT is the ground station identity. This value must be an integer.
* SCENARIO is the scenario to simulate. Must be an integer.
* DBHOST is the host where the data base is located. Must be a hostname or an IP direction.