SpaceOps-2023, ID # 1725

Implementing the CCSDS Service Management Interface at GSOC – Challenges, Obstacles and Considerations

Marcin Gnat[a]*, Wolfgang Frase[b], Edoardo Barbieri[c], Pierre-Alexis Lagadrillière[d]

[a] *Communication and Ground Stations, German Space Operations Center, German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany, marcin.gnat@dlr.de*
[b] *German Space Operations Center, Freelancer at Hays AG, Oberpfaffenhofen, 82234 Wessling, Germany, wolfgang.frase@dlr.de*
[c] *German Space Operations Center, LSE Space GmbH, Oberpfaffenhofen, 82234 Wessling, Germany, edoardo.barbieri@lsespace.de*
[d] *Communication and Ground Stations, German Space Operations Center, German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany, pierre.lagadrilliere@dlr.de*
* Corresponding Author

Abstract

A good plan is indispensable for successful space operations. This applies to the mission as a whole as well as the mundane things like the scheduling of ground station contacts. The planning of the ground station is an ancillary task that is carried out by any organization possessing antenna assets. Depending on antenna usage, number of assets and also even simple things like (missing) resources, each of these systems are different, yet they do their job. Interoperability between these operational systems can at best be described as difficult, as it suffers from many different interfaces and in the end reminds one of earlier times when non-standardized data protocols for telemetry exchange were an issue. In many cases, scheduling interoperability is achieved simply through human interaction, primarily by sharing of all necessary information via e-mail. However, e-mail is not perfect, especially with hybrid solutions. An organization that has an operationally automated system and communicates with another organization that is still planning on a human basis often has problems bringing these two worlds together. The CCSDS recognized the problem long ago and published the first issue of "Service Management" in 2009. However, the complexity of the issue and the solution presented prevented the actual implementation in most organizations. CCSDS has therefore re-started the project and is now working on "Extensible Service Management", which publishes recommended standards in slices that allow agencies and institutions to implement them more easily. Due to the fact that DLR is actively involved in this CCSDS activity, and the growing need for a new ground station planning system in general, we started a few years ago to redevelop our own system, which inherently supports new concepts that will be implemented in parallel to the development of CCSDS standards. Since no one is yet actively using the CCSDS interface, we decided to implement all legacy interfaces, one after the other, to facilitate integration into the operational environment in the coming months. Now that we are in the alpha phase of the software, we are forging some of these interfaces and, not really to our surprise, we are seeing many inconsistencies. Certain strategies, that seem to work internally for a single organization where software procedures are complemented by human interaction, do not comply with full automation and upcoming CCSDS standards, making it difficult to bridge the gap between new standards and the old world. The decisions that have to be made for things to work are not just a matter of software. They often have very profound operational implications and are deeply tied to all aspects of the mission. In our paper we present this new development at a high level and use it here as a basis for the analysis of current issues caused by what we call "interface matching". We show which inconsistencies we have been confronted with, which solutions or workarounds we want to implement, and we dedicate some attention to their effects on operations. Finally, we will discuss in generic terms several challenges and proposed solutions for the development of the modern ground station scheduling system, meant to remain maintainable for at least the next decade.

Keywords: ground station, scheduling, interfaces, service management, CCSDS

Acronyms/Abbreviations

Consultative Commetee for Space Data Systems (CCSDS), German Space Operations Center (GSOC), Deutsches Zentrum für Luft- und Raumfahrt (DLR), Low Earth Orbit (LEO), Geosynchronous Earth Orbit (GEO), Space Link Extension (SLE), Cross Support Service Management (CSSM), Deutsches Fernerkundungsdatenzentrum (DFD), Earth Observation Center (EOC), Kongsberg Satellite Services (KSAT), Simple Object Access Protocol (SOAP), Representational State Transfer (REST), Universal Time Coordinated (UTC), Equivalent Isotropic Radiated Power

(EIRP), Interface Control Document (ICD), Hypertext Transfer Protocol Secure (HTTPS), Ground Station Network (GSN), Acquisition Of Signal (AOS), Loss Of Signal (LOS), Extensible Markup Language (XML), Transmission Control Protocol (TCP), Internet Protocol (IP), JavaScript Object Notation (JSON), International Organisation for Standardisation (ISO), Two Line Element (TLE), File Transfer Protocol (FTP), Optical Ground Station (OGS)

1. Introduction

Satellite operations require a lot of coordination tasks with multiple aspects. We want to approach a specific case of such - the planning and booking of ground stations, popularly called ground station scheduling or station scheduling. To scope this term a bit more, we describe shortly the procedure.

There are space missions or a spacecraft, whereas they may be used as synonyms or be completely different things, where for example a space mission may have wide objectives and involve several spacecraft or other space or ground assets. There is also a mission owner or operator, which again may be the same entity, but also may be different ones, taking care of different aspects and parts of the whole setup. And to complicate things even more, there may be mission owner with separate mission operator, using several assets (i.e. spacecraft) with separate owners and yet separate operators. If we add to that the ground station providers (which in turn are yet different organizations) we get a pretty complex enterprise structure, as presented in Fig. 1.

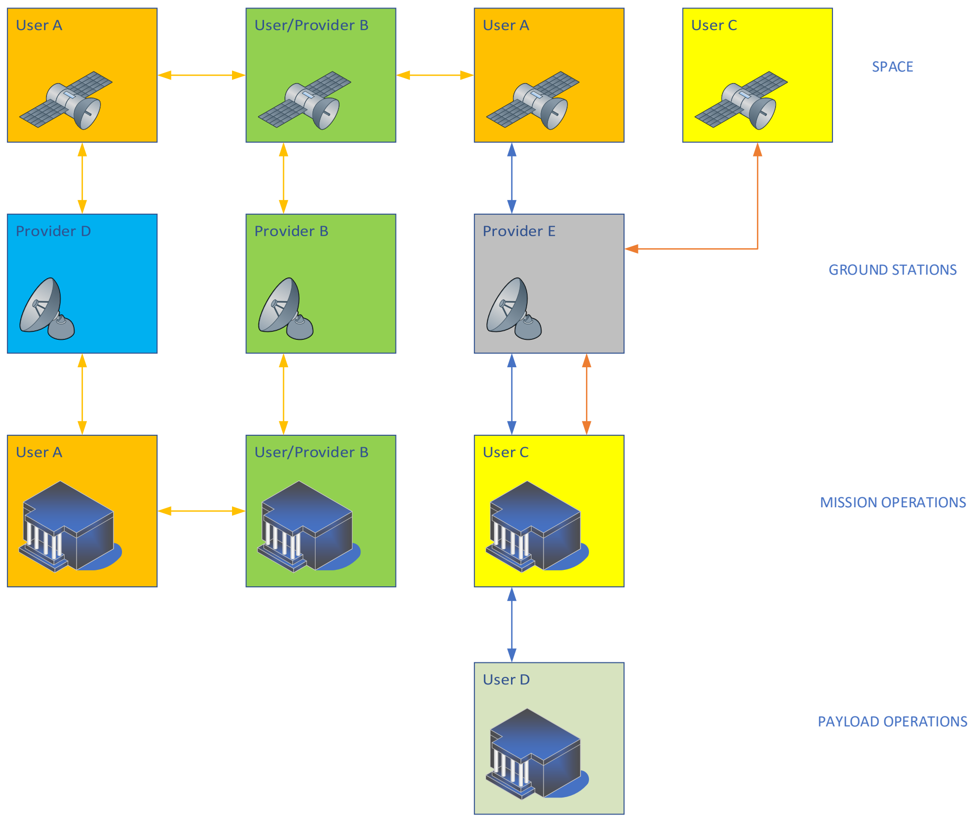

Fig. 1. Example complex enterprise structure

Now, let's imagine a way simpler scenario, as shown in Fig. 2. To continue from here, we will stick to the nomenclature as described in CCSDS SCCS Architecture Requirements [1]. We see there a spacecraft operated by an entity A and ground station service provided by the provider B.

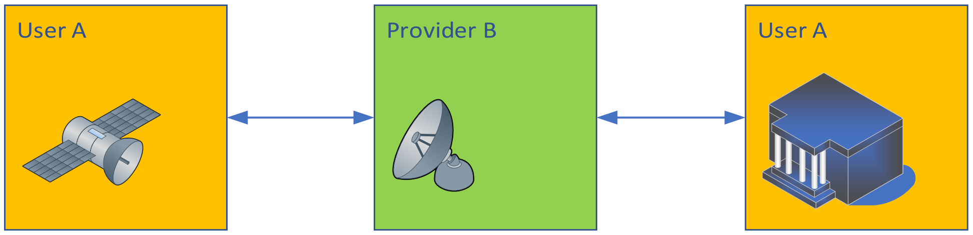

Fig. 2. Example simple ABA architecture

As for doing their operations, the operator A (which we may also call service user) needs to know where its spacecraft is located in space (orbit information) and possibly also wants to know, what the relative position to the specific ground station is. The orbit information is provided by a flight dynamics service (either located within the operator entity or used as external service). The contact window opportunity to the ground station is typically expressed by so called visibilities, means two-time events, which define where the spacecraft (within specific orbit) will start to be visible from the ground station and where it won't be any more. In case of LEO the time period where the spacecraft is visible may span between 3 to 10 minutes. For GEO spacecraft there is permanent visibility. In case of deep space missions, we may have several hours of visibility, limited only by the actual Earth rotation (where station "moves" away from the far spacecraft to the back side of the Earth). In any case such visibilities are not very hard to calculate, especially when no high precision is required (typically precision within a minute is enough). Therefore, such calculations may be performed by flight dynamics services, locally by an operator, or the station provider. In any case, such visibility predictions are of high importance for the operator (its mission planning service needs to plan the on-board timeline for data dumps for example) and for the station provider (to assess availability of the station resources).

This is the moment where the so-called station scheduling takes effect. In general, it is a process which takes care about the coordination and booking of spacecraft-to-ground contacts based on aforementioned visibilities and any additional constraints existing at participating parties. The operator (service user) defines its request to the provider, either in form of specific contacts or asking for provision of contacts based on some constraints. The station provider analyses the request and delivers back to the user a response, containing respectively confirmed (or rejected) passes or a contacts proposal based on user constraints. This typically needs to be done in specific time prior the actual pass, giving time to both parties to prepare operations (generate timeline) on one side and reserve resources (book station times) on another. The process as described is shown on Fig 3.

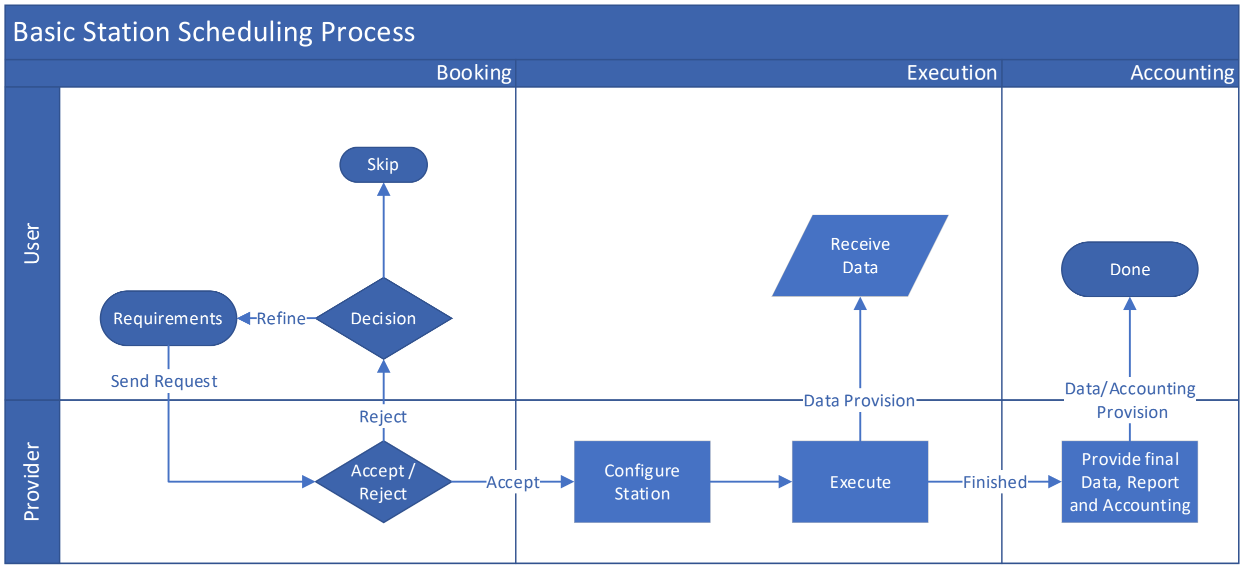

Fig. 3. Basic scheduling process

As the space systems and missions are complex and operate in even more complex environment, it is not unusual this scheduled setup is being changed - passes are being added or removed, short term contingency bookings come into the game or resources are becoming unavailable, just to mention a few of possible distortions. Our simple process as presented above suddenly gets very complicated (Fig 4) and the example shows just few of possible distraction paths. Almost at any place one can imagine further injection of new requirements, unavailability, cancellations and what else, where the number of possible distortion sources and their implications is almost infinite.

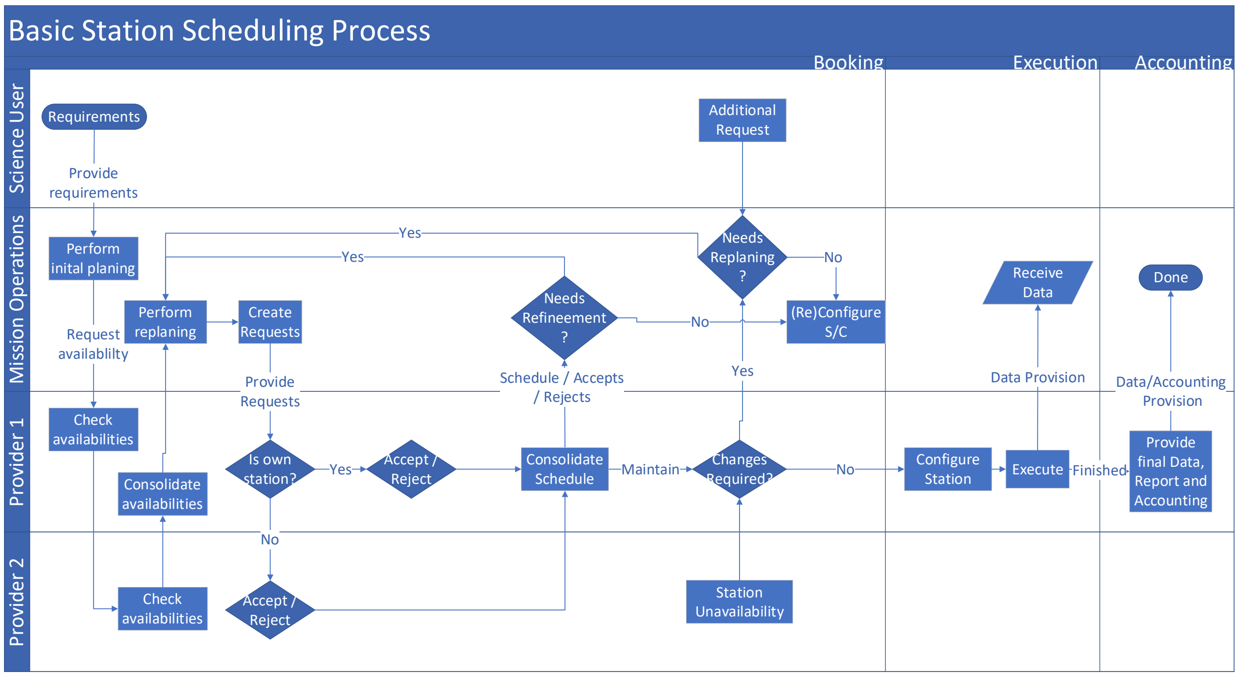

Fig. 4. More complex scheduling process

Every separate organization tends to set up and use their own procedures and interfaces. Currently there are some basic communications standards used worldwide, like TCP/IP, which improve operations significantly, but just one level above in the protocol stack and it does not keep up to modern standards. For a lot of specific space related data like telemetry, orbit predictions and the here discussed scheduling, there are at least as many different versions of implementations as organizations. There are some efforts to streamline this situation, especially within CCSDS, and there are some successes like the SLE [2], but it’s only one drop of water on a hot stone. Still one can see a rule: “I provide a service thus I provide an interface”, sometimes even “I want a service and I want a special interface”, which quickly ends in “n-to-n” interfaces in the wild. Now just doing a simple math, multiplying this with the actual scheduling process complexity described above, yields a real problem.

Of course, the whole space business would not work if we would just state that. What actually happens is that a lot of things are being extremely simplified. The enterprise model is being limited to a basic one (ABA configuration), and the scheduling process is also stripped to the basics. In case of any specialities, inconsistencies and issues one uses an ultimate universal intelligent system with associative database and predictive decision making – the human. Not surprisingly, humans can cope pretty well with all possible (and impossible as well, in fact) problems arising. What humans do not do very well, is to perform hundreds, thousands and millions of small, repetitive, precise operations, like checking a pass request against the existing database with hundreds of entries. And you may see already, as long as the space operations were singular and exclusive, it was pretty simple to involve humans to support such scheduling and coordination tasks. Now however, with increasing number of ground antennas [22,23,24,25] and spacecrafts like SpaceX’s Starlink fleet [3], we reach the areas where human errors may impact the operations. Automation is required, to keep up with the number of involved components in the system and at the same time reduce the cost of operation. Especially in so called cross-support landscape there is a need for standard interfaces that cover at least the majority of the space operational scenarios and enterprises.

The Cross-Support Services (CSS) are being provided in scope of previously mentioned enterprise model mainly in form of the SLE. The SLE Services focus on actual spacecraft commands and telemetry provision on a terrestrial links, allowing for some flexibility if using specific services (like Return All Frames [4] vs. Return Channel Frames [5]) or their options (timely vs complete data provision). One of the aspects of these services is their configuration. Providing wide flexibility requires also configurability. And the SLE, being a very robust handshake protocol, needs proper configuration beforehand. This configuration needs to be provided out-of-band. The SLE standard actually foresees upcoming Service Management standards, which shall take care of this aspect.

It took several years to develop the first CCSDS Cross Support Service Management (CSSM) standard in 2009 [6]. The standard was holistically approaching the Service Management, trying to cope with service and information identification, conveyed information, detailed exchange patterns and binding to the underlying protocol (in this case SOAP). Although very thoroughly conceived, it did not really appeal to most of the agencies and companies. It was seen as too complex, implementation intensive and not really worth replacing relatively simple e-mail-based booking processes, which were widely spread that time. And so, the standard existed for over 10 years, without a single inter-agency implementation.

CCSDS recognised that the complexity not necessarily solves the problem. Therefore, it has been decided to restart the development of the standard with a different approach. The overall Service Management landscape is still far away from being simplistic, and digging into all details shows a good amount of complexity, but rather due to the complexity of the issue – station resource booking - itself. The idea was, however, to take that complexity away from the actual implementation and the interface, leaving it for later efforts, or giving implementers options to expand the interface later on, depending on own system evolvement and needs.

New extensible Cross Support Service Management (CSSM) is thus organised around the so-called mission support lifecycle of the booking process (like depicted in Fig. 5 below, from [7]). It has been recognised, that at each step of the lifecycle, specific exchange between mission user and provider takes place, which can be reasonably distinguished from other steps. Thus, the CSSM is actually a collection of several standards, which are individually implementable and can be step by step integrated into existing systems.

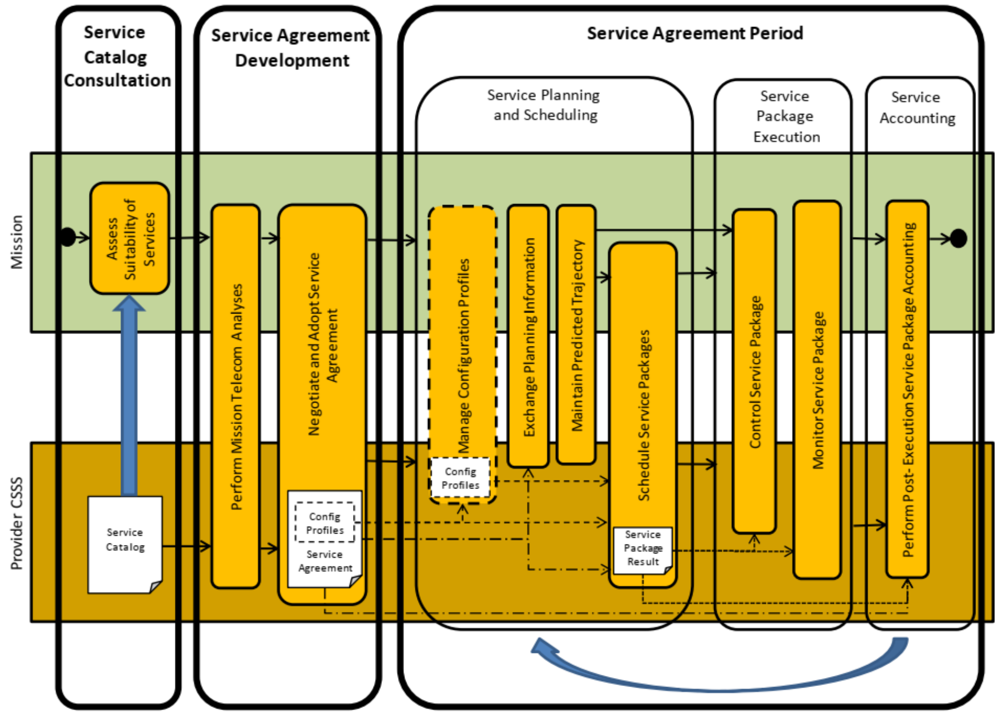
Service Catalog Consultation
Service Agreement Development
Service Agreement Period
Service Planning and Scheduling
Service Package Execution
Service Accounting
Mission
Provider CSSS
Assess Suitability of Services
Service Catalog
Perform Mission Telecom Analyses
Negotiate and Adopt Service Agreement
Config Profiles
Service Agreement
Manage Configuration Profiles
Config Profiles
Exchange Planning Information
Maintain Predicted Trajectory
Schedule Service Packages
Service Package Result
Control Service Package
Monitor Service Package
Perform Post-Execution Service Package Accounting

Fig. 5 CSSM Mission Support Lifecycle

CSSM consists of standards defining so called information entities, which in essence define formats for specific information pieces used during station booking process. Additionally, there are a few ancillary books, with common data definitions. There is also the actual standard defining possible automation of the process, using some exchange protocol, will be defined last, considering the hoped usage of individual, already released, standards for the information formats. The already existing and shortly upcoming standards are:

- Simple Schedule Format (SSF, [8]) providing the overall schedule information of reservations made at a provider,

- Communication Planning Information Format (CPIF, [9]), providing event-based information of theoretical plan of support
- Service Management Utilization Request Formats (SMURF, [10]), allowing mission user to request multiple actions at the provider, including actual booking requests
- Service Package Data Format (SPDF, [11]), providing consolidated information set for an upcoming service to be executed by provider

Still in development and upcoming in next few years are standards for:

- Service Agreement and Configuration Profile Formats (SACP, [12]), allowing provision and exchange of a spacecraft and ground configuration to allow automated set up of all assets and service execution
- Space Link Event Sequence Format (EVSQ, [13]), allowing provision of the detailed event sequence during service provision
- Service Accounting (SACC, [14]), providing post service information on received data amounts and overall quality of the service
- Management Services (SMMS, [15]), providing exchange protocol and details on how to involve service identification, state changes and automation.

The Service Management considers also the so-called functional resources, which in turn allow more complex control of the resource allocation for the space mission. This means, automated booking and station configuration at the same time becomes possible, whereas these two things were treated independently from each other until now. This feature is however treated as optional, and thus can be implemented in later stage.

Another aspect addressed by CSSM is the unique and inter-agency valid identification of different information types, which starts from respective message/information entity design (each service package, each trajectory, etc… have their unique identifiers) and through usage of central registry – SANA [16] for unified storage and provision of existing identifiers for multiple uses (for example most known spacecraft IDs, but now also upcoming station and antenna IDs, resource object identifiers registration for configuration parameters, and few more).

The challenge which especially attracted our attention was the actual implementation of the interfaces for scheduling and station configuration. In the next section we present shortly the concepts of the new CCSDS Service Management, which are the basis for our new development, followed by short description of the actual software architecture implementation, and finally discussing the issues and solutions which we implement to cope with interface mismatches.

2. GSSNG - GSOC Implementation

The Ground Station Scheduling Next Generation (GSSNG) was born around six years ago as an idea to renew GSOC's scheduling landscape. It was meant not only to replace the software that was until now in operation, but mainly enable new interfaces, and allow changing complete processes around scheduling operations. Ultimately, it will allow full automation of ground station planning, scheduling, and execution of satellite passes.

Except for the obvious task of station scheduling, the GSSNG has been designed around following central points: maximum support for emerging CCSDS SM standards, implementation of legacy interfaces and integration into GSOC infrastructure while complying to the internal security policies.

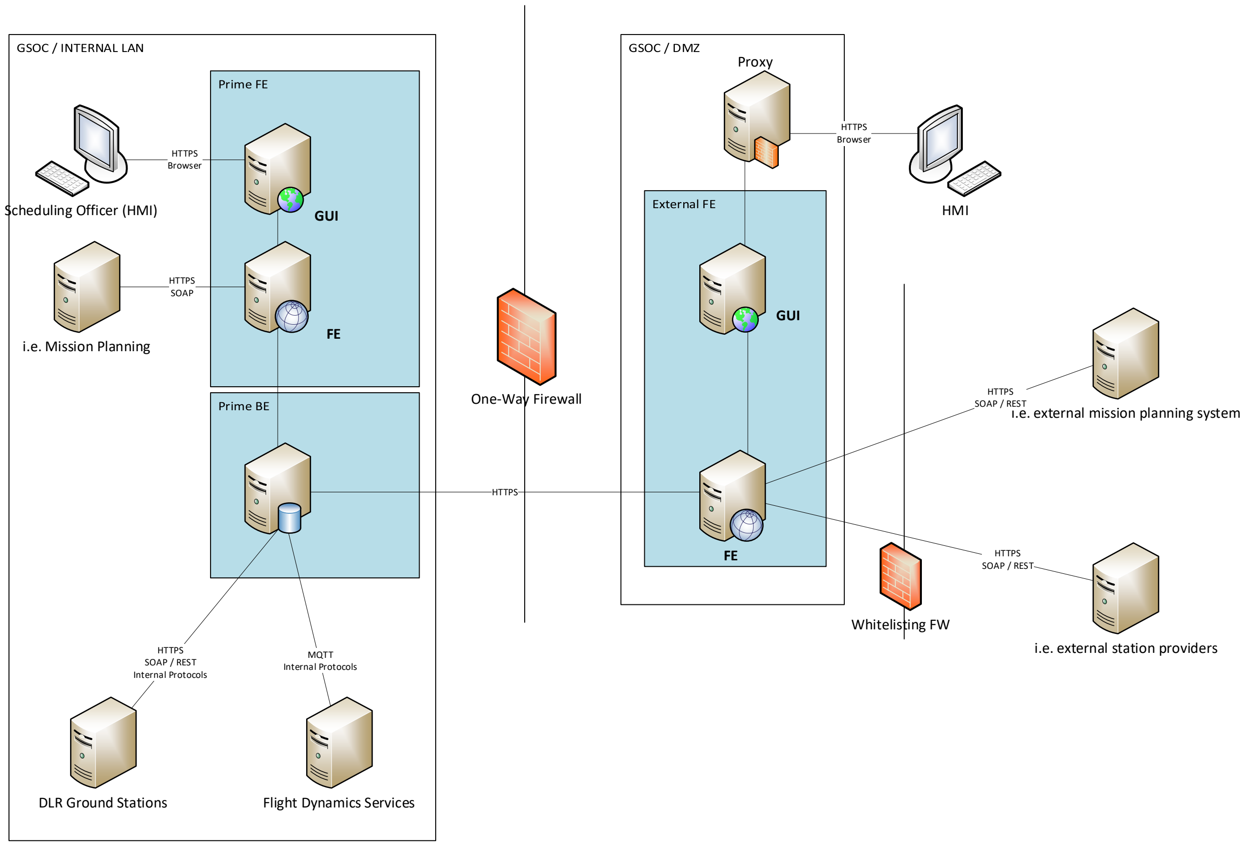

Fig. 6 GSSNG System Overview

The Fig. 6 shows the architecture of the system. For simplicity, only primary systems are shown, but in principle every one of them is intended to have a hot backup sibling running in the background. The system takes an advantage of dividing the components into backend (BE) and frontend (FE), where BE is the main scheduling server containing the system logic and the database. The FE acts as interface provider and is the only component which is exposed to the outside connections. FE contains plugins to each interface type supported, thus it can handle different interfaces at the same time. The BE however shares these plugins, as in most of the cases there is some additional processing needed, and the access to the database as well as the business logic are not available at the FE. There is actually a third component which we call GUI, but for simplicity, the GUI runs on the same physical server as FE. The GUI is nothing else as a server for providing web access to the human users.

To fulfil the security requirements (like exact tracking of systems working in an operational environment, firewall rules), the FE components (which may be multiple instances connected to a single Backend) need to be registered at the BE by the system administrator. Through whitelisting (IP address and SSL certificate) we guarantee that only desired FE machines are communicating with BE. Another speciality in case of communication outside of GSOC is that we need to traverse the firewall. As the firewall does not allow any connections from outside into the GSOC network, the BE actually uses a polling technique [17] to communicate with the exposed FE.

The complete communication between BE, FE and GUI was originally intended to be performed using CCSDS defined standard information entities. However quickly we've had to overhaul these formats, as there are some – maybe not big, but important – auxiliary information which we need to exchange. For example, actual authentication and authorisation within the system, additional operations on the database and in general user and user management related actions needed to be extended. Therefore, there is a large similarity between GSSNG and CCSDS data entities, and GSSNG can be seen as encapsulating the actual CCSDS information where appropriate.

3. Inconsistencies of Interfaces

As already mentioned, one of the main reasons behind the new development was the support of various interfaces between GSOC and external partners. Historically most of the booking processes were performed via e-mail, in more or less formalized way. Despite the CCSDS efforts described in introduction, the standardized interface did not establish itself yet. Nevertheless, the complexity of missions increased and also simply the volume of supports just exploded. Many of our partners developed thus their own interfaces, unfortunately with almost no larger scale coordination. Everybody started to develop some ideas, mainly of course matching their own needs. This is now the reason that practically every single interface coping with station booking we use, is different. In fact, in comparison to the previous e-mail-based exchange and just a human user who analysed and implemented the information, it is a large step backwards, as it causes additional efforts and even unintended crashes of external automated systems, if the information does not arrive on time or in right format.

Luckily until now most of the differences were based on the file format differences, and we did not have to cope with actual protocols. Files are being send by e-mail as an attachment or provided via FTP. The file formats divided until now into XML and plain text based. Especially the plain text format is tricky, as the information inside is organised in a very custom way as comma or space or tab separated values, forming kind of tables or semi-tables, combined with free text (especially with respect to comments).

Both formatting ways showed us in the past that on our implementation side we need to cover for many specific cases. For the XML it is a bit easier, as the XML definition itself gives a pretty good framework for the information content. Nevertheless, we noticed that handling of XML was not always perfect, and many implementations lacked good XML schemas or actually misused or, in the best case, ignored best practices of XML world. In case of plain text-based formats, the parser needs to find the structure in the file, what kind of character being used as a separator to allow to slice the information in right way. And already existing parser could also get irritated by simple end of line (EOL) conversion between Windows and Unix ones.

Nowadays we observe increased usage of REST architecture for near-real time exchange and fully automated systems. As it seems to be the protocol of choice also for future CCSDS implementations, it shall not pose an issue, however, and we do not cover that further in this paper.

Final, and the most problematic, inconsistencies are imposed actually by the information content itself or actually missing of one. It started to be very clearly apparent first when we used software systems to parse or produce information. Many components previously just accepted by human operators, suddenly cause issues which render the interfaces and their implementation, especially, very complicated. The problems start already with basic things like a definition of “pass”. What is a “pass”? When does it actually start? Is the pass which was interrupted still the same pass or another one? How to treat handovers? How to provide information on configuration, and when? What does “configuration” mean at all? Things which were accepted before, because “we did it always like that”, are not anymore easily implementable, as software systems require more clarity and standardization.

Another major issue we found is the actual identification of the booked contacts. What was again very easy for a human (“could you please delete the pass at 11:12am?”) is almost impossible for automated software system. How should the system recognise which service it is if times are defined with precision up to seconds? The information provided in a request like above is just not enough. We could of course try to accommodate for that, and introduce validity windows, but still the uncertainty stays. We can also ask the requestor to provide precise time, but when it is just off by 1 second, is it valid, or not? And was it at all time provided as local time or UTC? All that multiplies with increasing number of missions and antennas used.

To better present the issues, we have pulled three existing interfaces which are being currently used daily in our operations. These are GSOC legacy interface, KSAT legacy interface and DLR’s DFD interface. In comparison we try to show upcoming CCSDS CSSM advantages.

3.1 Legacy GSOC Interface

The GSOC legacy interface is a collection of different formats used for different reasons over the span of several years. Unfortunately, more emphasise was put on compatibility with existing systems than having unified interface, thus the different formats are largely incompatible with each other and require considerable converting effort.

The main inputs of the scheduling software are files in the XML or text format. These inputs are generated and delivered by each project, in order to request passes for any ground station in their ground stations network. These inputs were manually imported by the GSOC Scheduling Office. The output of the scheduling software is the result of processing the projects inputs and the ground stations availabilities.

The formats used by GSOC legacy system consist of plain text-based request and schedule, XML based request and schedule, as well as more or less formalised automated mail notifications send around.

The text schedule request is mainly used for sending requests to external agencies and ground stations, via e-mail. The XML request have been used by internal projects (like input from mission planning) to our scheduling system as well as for several external supported projects (mostly university satellites).

The XML and text published schedule outputs are the released schedule formats for a given week. The complete XML file is the actual interface to Mission-Planning at GSOC, where it is used to generate onboard timeline for station contacts. An Email notification is automatically generated by the software for each new released version of the schedule.

There is no unique contact identification, this is realised by matching AOS and LOS times. Also, in case of schedules, each item is being represented by two lines, where one includes actual tracking pass and another one the general activity times around it. A special case is very different and very nonstandard handling of date and time in files.

```
DATE         SAT     STN         REVN    MAX.ELE    AOS              LOS         OPRN

23 01 09     EN1     SG6         4170    72.6       13:29:45         13:40:42    S
23 01 09     EN1     SG6         4171    55.3       15:06:18         15:17:05    S
23 01 10     EN1     SG6         4182    23.2       09:03:59         09:13:39    S
23 01 10     EN1     SG6         4185    66.1       13:54:46         14:05:41    S
23 01 10     EN1     SG6         4186    53.9       15:31:16         15:42:02    S
23 01 11     EN1     SG6         4197    27.2       09:29:08         09:39:10    S
```

Fig. 7 GSOC Legacy text request

```
Monday   PERIOD: 23.01.09 - 23.01.16    DTG OF REV: 23.01.02 / 0810utc     REV:  0
REMARKS: Draft

==  ===  ===  ========  =====  =====  ==========  =============  =============================================  ======  ===
CW  DOY  DOW  DATE      START  STOP   SUP ID      STATIONS       ACTIVITY                                       ORBIT   ITM
==  ===  ===  ========  =====  =====  ==========  =============  =============================================  ======  ===
 2    9  MON  23.01.09  0000z  2400z  TX1         OHIG/OHG       SPT                                         .  0       1
 2    9  MON  23.01.09  0002z  0037z  TD1         INUV/INU       SPT                                         .  0       2
 2    9  MON  23.01.09  0022z  0032z  TD1         INU            PASS                                       P-  69608   3
 2    9  MON  23.01.09  0059z  0132z  TD1         OHIG/OHG       SPT U/L STBY                                .  0       4
 2    9  MON  23.01.09  0119z  0127z  TD1         OHG            PASS U/L STBY                              P-  69608   5
 2    9  MON  23.01.09  0135z  0211z  TX1         INUV/INU       SPT U/L STBY                                .  0       6
 2    9  MON  23.01.09  0146z  0217z  TX1         KSAT/SG6       SPT X                                       .  0       7
 2    9  MON  23.01.09  0155z  0206z  TX1         INU            PASS U/L STBY                              P-  86339   8
 2    9  MON  23.01.09  0206z  0212z  TX1         SG6            PASS X                                     P-  86339   9
 2    9  MON  23.01.09  0311z  0345z  TD1         INUV/INU       SPT U/L STBY                                .  0       10
 2    9  MON  23.01.09  0320z  0354z  TX1         KSAT/SG6       SPT X                                       .  0       11
 2    9  MON  23.01.09  0331z  0340z  TD1         INU            PASS U/L STBY                              P-  69610   12
 2    9  MON  23.01.09  0338z  0412z  GF2         NST/NSG        SPT                                         .  0       13
 2    9  MON  23.01.09  0340z  0349z  TX1         SG6            PASS X                                     P-  86340   14
 2    9  MON  23.01.09  0358z  0407z  GF2         NSG            PASS                                       --  25776   15
 2    9  MON  23.01.09  0500z  0536z  TX1         NST/NSG        SPT                                         .  0       16
 2    9  MON  23.01.09  0502z  0538z  TD1         WHM/S69        SPT DUAL                                    .  0       17
 2    9  MON  23.01.09  0502z  0538z  TX1         WHM/S69        SPT DUAL D/L ONLY                           .  0       18
 2    9  MON  23.01.09  0520z  0531z  TX1         NSG            PASS                                       P-  86341   19
 2    9  MON  23.01.09  0522z  0533z  TD1         S69            PASS DUAL                                  P-  69611   20
```

Fig. 8 GSOC Legacy text schedule

```
<?xml version="1.0" encoding="utf-8"?>
<requestList comment="str111" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="schedule_request.xsd">
          <request comment="str111">
                    <pass>
                              <aosDate>2015-12-13</aosDate>
                              <aosTime>12:02:14</aosTime>
                              <losDate>2015-12-13</losDate>
                              <losTime>12:12:12</losTime>
                              <satellite>FLP</satellite>
                              <station>WHM</station>
                              <orbitNumber>33</orbitNumber>
                              <maxElevation>67.25</maxElevation>
                              <dataRate>H</dataRate>
                              <passPriority>P</passPriority>
                              <requestPriority>Prime</requestPriority>
                              <activity>S D/L ONLY</activity>
                    </pass>
          </request>
</requestList>
```

Fig. 9 GSOC Legacy XML request

```xml
<?xml version="1.0" encoding="UTF-8"?>

<schedule_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="schedule.xsd" generated_date="2023-01-
02T08:10:47.938Z" entries_from="2023-01-09T00:00:00.00" entries_until="2023-01-16T00:00:00.00">

<schedule_entry  priority="-" station="OHIG/OHG" stop_time="2023-01-09T00:00:37.00" start_time="2023-01-08T23:25:02.00"
                 support_id="TX1" activity="SPT" data_rate="-" is_pass="false" ></schedule_entry>
<schedule_entry  priority="-" station="INUV/INU" stop_time="2023-01-09T00:37:44.00" start_time="2023-01-09T00:02:20.00"
                 support_id="TD1" activity="SPT" data_rate="-" is_pass="false" ></schedule_entry>
<schedule_entry  max_elevation="34.7" priority="P" station="INU" stop_time="2023-01-09T00:32:44.00" start_time="2023-01-09T00:22:20.00"
                 support_id="TD1" activity="PASS" data_rate="-" is_pass="true" ></schedule_entry>
<schedule_entry  priority="-" station="OHIG/OHG" stop_time="2023-01-09T01:32:53.00" start_time="2023-01-09T00:59:32.00"
                 support_id="TD1" activity="SPT U/L STBY" data_rate="-" is_pass="false" ></schedule_entry>
<schedule_entry  max_elevation="11.5" priority="P" station="OHG" stop_time="2023-01-09T01:27:53.00" start_time="2023-01-09T01:19:32.00"
                 support_id="TD1" activity="PASS U/L STBY" data_rate="-" is_pass="true" ></schedule_entry>
<schedule_entry  priority="-" station="INUV/INU" stop_time="2023-01-09T02:11:46.00" start_time="2023-01-09T01:35:55.00"
                 support_id="TX1" activity="SPT U/L STBY" data_rate="-" is_pass="false" ></schedule_entry>
<schedule_entry  priority="-" station="KSAT/SG6" stop_time="2023-01-09T02:17:30.00" start_time="2023-01-09T01:46:40.00"
                 support_id="TX1" activity="SPT X" data_rate="-" is_pass="false" ></schedule_entry>
<schedule_entry  max_elevation="65.7" priority="P" station="INU" stop_time="2023-01-09T02:06:46.00" start_time="2023-01-09T01:55:55.00"
                 support_id="TX1" activity="PASS U/L STBY" data_rate="-" is_pass="true" ></schedule_entry>
```

Fig. 10 GSOC Legacy XML schedule

3.2 KSAT Legacy Interface

The scheduling interface is defined through different XML messages exchanged between the customer and KSAT. Three different XML messages constitute the scheduling interface:

- schedule request message
- schedule request reception confirmation message
- schedule reply message

The different formats are described in the [18]. Contacts which are being deleted do not get explicit deletion confirmation, but rather are not present in schedule reply message at all (confirmation by absence).

The number of schedule reply messages will vary dependent on the type of scheduling. In case of nominal scheduling there will only be one schedule reply message sent after deadline, regardless of how many schedule request messages has been sent by the customer. For short term and emergency schedule request, one schedule reply will be sent per schedule request received.

The schedule reply message contains the status of all requested passes for all satellites requested by the customer. KSAT uses AOS and LOS to uniquely identify different passes.

In order to get an overview of the available time slots on the KSAT antenna system, before sending a schedule request, the customer may use the acquisition availability interface described in the document [19]. This interface consists of the two XML messages, the acquisition availability request message and the acquisition availability reply message. Based on the information in the acquisition availability message, the customer selects passes to request, generates and sends a schedule request message to KSAT asking for the free time slots of interest.

```
<?xml version="1.0" encoding="UTF-8"?>
<ksat_schedule_request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLoca-
tion="http://cweb.ksat.no/cweb/schema/schedule/schema_schedule_re quest.xsd">
        <header>
                <originator>MySatelliteCompany</originator>
                <recipient>KSAT</ recipient>
                <schedule_valid_from>2008-12-17T00:00:00Z</schedule_valid_from>
                <schedule_valid_to>2008-12-18T00:00:07Z</schedule_valid_to>
                <request_reference>HGT4285T3</request_reference>
                <generation_time>2008-12-16T08:24:00Z</generation_time>
        </header>
        <body>
                <schedule_request>
                <request_id>545164</request_id>
                <action>ADD</action>
                <start_time>2008-12-17T09:30:47Z</start_time>
                <end_time>2008-12-17T09:33:47Z</end_time>
                <satellite_name>MySat</satellite_name>
                <requested_antenna>SG3</requested_antenna>
                <uplink>NO</uplink>
                <configuration>
                        <parameter>
                                <name>StartOffsetTime</name>
                                <value>60</value>
                        </parameter>
                </configuration>
                </schedule_request>
        </body>
</ksat_schedule_request>
```

Fig. 11 KSAT Schedule request example.

```
<?xml version="1.0" encoding="UTF-8"?>
<ksat_schedule_reply xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLoca-
tion="http://cweb.ksat.no/cweb/schema/schedule/schema_schedule_re ply.xsd">
        <header>
                <originator>KSAT</originator>
                <recipient>MySatelliteCompany</recipient>
                <schedule_valid_from>2008-12-17T00:00:00Z</schedule_valid_from>
                <schedule_valid_to>2008-12-18T00:00:07Z</schedule_valid_to>
                <request_reference>HGT4285T3</request_reference>
                <generation_time>2008-12-16T16:00:00Z</generation_time>
        </header>
        <body>
                <schedule_reply>
                        <start_time>2008-12-17T09:30:47Z</start_time>
                        <end_time>2008-12-17T09:33:47Z</end_time>
                        <satellite_name>MySat</satellite_name>
                        <antenna>TG2</antenna>
                        <orbit_number>45483</orbit_number>
                        <status>ACCEPTED</status>
                </schedule_reply>
        </body>
</ksat_schedule_reply>
```

Fig. 12 KSAT Schedule response example.

3.3 DFD Interface

DFD scheduling interface uses a number of plain-text files with comma separated values [20]. A schedule request conveys the wish of the user to use the system during a specific period of time. The request includes information on requested antenna, times, priority, configuration and mission tag.

Lines containing only Space, CR and LF characters are permitted, but ignored. Lines beginning with a '#' character are ignored (are treated as comments), fields are separated by comma, spaces are not allowed.

```
SSC,Support4711,INU,ADD,6,2011-11-20T16:12,2011-11-20T16:24,SSC_INU_CFG,TT&C
#This is a request DLR,Eng5,IVK,ADD,9,2011-11-27T03:30,2011-11-20T05:30,RNG_LOOP_CFG,"Loop Test"
```

Fig. 13 DFD text request

The antenna plan (resource plan) is a DFD result of planning and scheduling decisions. It contains the information on the status of the related request, the resource to which the request relates, time frame for which the request shall be effective, the priority of the request, configuration that shall be used, identification of the user, dentification for use by user (mission tag), and the comment.

Interesting to note here is handling the schedule plan differently for end customers (clients) and partner networks. Partners get an antenna plan that contains all scheduling requests. The request times in this plan refer to the allocation of the system. Clients get an antenna plan that only contains their requests. The request times in a client plan refer to the requested service time.

Format of the schedule is very similar to the request format, comma separated values in plain-text.

```
Example of a client antenna plan:
# INU Antenna Plan generated at 2011-11-20T16:45:03
# Format:
<ORIGIN>,<MISSION_TAG>,<RESOURCE>,<STATE>,<PRIORITY>,<STARTTIME>,<STOPTIME>,<CONFIG>,<COMMENT>
#
MDA,RS2_074345,INU,SCHEDULED,5,2011-03-26T02:40,2011-11-27T02:43,RADARSAT2_CH2,
MDA,RS2_074346,INU,DELETED,5,2011-03-26T04:15,2011-11-27T04:24,RADARSAT2_CH1,
MDA,RS1_5546A000753,INU,SCHEDULED,5,2011-03-26T05:03,2011-11-27T05:13,RADARSAT1,
MDA,RS1_5568A000945,INU,PLANNED,6,2011-11-27T13:02,2011-11-27T13:08,RADARSAT1,
```

Fig. 14 DFD text request

In order to help missions to utilize the station, the GSN can provide a number of planning support files. The generation and delivery of those products can be configured on a per mission base. Planning support files are generated on a regular basis and reflect visibility and scheduling constraints as known at the time of generation. The files govern the nominal request time window. Mission tailored products only give information for those times during which the mission has nominal1 visibility. In effect this hides scheduling constraints that are not effective for the mission.

3.4 Summary of differences

In the below table we summarize the differences between the abovementioned interfaces and formats. For a comparison we provide a column with CCSDS Cross Support Service Management. Please note, the table is not exhaustive and there are definitely other interfaces which may have selected parameters similar to CCSDS. The intention is to show general issue and not to focus on any specific implementations and their advantages or disadvantages.

Table 1. Comparison of selected scheduling interfaces

	GSOC Legacy	**DFD**	**KSAT Legacy**	**CCSDS CSSM**
Communication Medium	E-Mail	FTP	FTP	FTP, HTTPS/REST
Information	Loose Text, File	File	File	File, Message
File Format	ASCII, XML	ASCII	XML	XML
Unique Service (request) Identification	No (time only)	Partially (mission name and orbit number)	No (time only)	Yes
State/Status Information	No	Yes	Rudimentary	Yes
Human Interaction Required	Yes	No	Partially	No
Station Configuration Information Available	No (very rudimentary configuration name)	No (very rudimentary configuration name)	No (assumed it is known)	Yes
Distinct Confirmations to Requests	No (only by mail and indirect through schedule)	No (only by mail and indirect through schedule)	No (only by mail and indirect through schedule)	Yes
Station Availabilities Provided	No	Yes	Yes	Yes
Real-time Schedule Information	No	Yes (limited)	Yes (indirect)	Yes

From the above table it can be clearly derived the age of each interface, where especially the GSOC legacy one does not support modern or automated communication. It does not provide any explicit confirmations, nor the distinction between different operations (addition, deletion). It requires human interaction and also it does not provide any station availability information. The DFD and KSAT systems are one step further, even using different file formats, but at least allowing for availabilities, partial automation and providing rudimentary status information.

The CCSDS based interface takes advantages of modern real time REST based architecture running over HTTPS, but also inherently (and thus in case of file exchange over FTP) provides decent information content and states which allow full automation of the operations.

Going more into details, we can compare the service package (or pass) identification. In case of GSOC and KSAT legacy interfaces there is no dedicated identification, and both user and provider are obligated to identify passes just by their start and stop times. We discuss the issues of date and time in following chapter in more detail. Except for CCSDS, only DFD provider some kind of combined identification, which is constructed out of mission name and orbit number. In many cases it seemed enough, however one can clearly not guarantee uniqueness of mission name tag or correctness of orbit number, which in both cases can lead easily to mismatch.

The real time schedule information is a category which we selected, as it was often asked by users in our past surveys and complained about missing one in case of our old GSOC system. The GSOC Legacy system did not provide real-time schedule information, users had to rely on weekly releases. In case of DFD there is a possibility to see current schedule via dedicated (yet not available for everybody) web page, as well as antenna plan files, which can be picked by customers in near real-time. In case of KSAT, the schedule information can be requested, or will be generated new in case of changes. Newer systems (like CCSDS CSSM) inherently include operations for getting the current schedule on request of user at any time.

4. Interface Matching

For our development, we targeted CCSDS Service Management as main interface. As said before, it constitutes most of the decisions on database design and internal interfaces. However, due to the fact that the standard is not yet released nor even finished, we have decided for the operational use to support most important interfaces or file formats in general, in order to enable the system as soon as possible. During our development, we noticed pretty soon, that such integration of multiple, partially very different interfaces is complicated. There are several aspects of the interface, which are partially perpendicular to each other, and thus changes in one causes implications in another one. One of the examples could be something as trivial as a date or time format. On one hand, one may have a desire to make it more precise (for example including seconds or even milliseconds), doing this however very quickly leads to problems with scheduled pass identification, as some of the existing interfaces use date and time of the pass as its unique identifier (but only with the precision up to minutes). Another issue we had to cope with is general information content, and decisions we need to make on which parts of the provided information on one interface correspond to second interface, and if they need to be processed or can be forwarded as they are. Finally, one of the central aspects of our analysis focused on information states or, in general, their state machines. It showed up, that matching different state machines between interfaces is particularly tricky, as it actually influences the result of complete booking process.

When something does not match directly, there are few strategies to cope with that, in general. They can be divided in complex and simple solutions. Complex ones would be where you try to augment for missing or not matching information by trying to recognise what should be (with sometimes complicated algorithms or patterns). With this method you can almost recover from the interface mismatch, but on the cost of time invested in programming, as well as increasing an interface complexity. On the opposite side, you'd have a method, where you just ignore things or simplify the reaction. Things which are clear are matched; everything else is dropped in one common category. This is robust, but you get some kind of fuzziness for special cases, which may cost you an issue during operations.

4.1 Date and time

The format of time and date would seem to be something obvious, yet imposes some interesting issues. There are differences in how computer systems actually store date and time (i.e. in form of integer value of seconds since some arbitrary selected date) and how humans tend to think about and write it down. And when we speak humans, we also need to account for different geographical specialities, like the precedence of month before the day in US American date notation, whereas in Europe rather opposite is the case.

Luckily, there is an ISO standard for the date format [21]. In general, it covers most of our issues; however, it also leaves some flexibility, which in case of an operational data as used for space application, may also bring problems.

The below table collects several date and time formats used in general applications and in scheduling systems.

Table 2 Example Date and Time formats comparison

Format Source	Date and Time	Comment
ISO Standard	2020-02-24T14:35:00Z 2020-02-24T14:35:00.000Z 20200224T143500Z	Few more formats are actually defined
US "Typical"	02/24/2020 2:25pm	For comparison, as used in typical human communication.
European "Typical"	24.02.2020 14:35	For comparison, as used in typical human communication.
CCSDS Service Management	2020-054T14:35:00.000Z	CCSDS ASCII Time Code B
GSOC Legacy Text Schedule	20 02 24 14:35:00	Date and time are in separate fields/columns. Values require leading zeros.
GSOC Legacy Text Request	2020/02/24 14:35:00.0	Date and time are in separate fields/columns. Values require leading zeros.
GSOC Legacy XML Request	2020-02-24 14:35:00	Date and time have extra defined datatypes and are provided in separate fields.
GSOC Legacy XML Schedule	2020-02-24T14:35:00.00	Uses xsd:dateTime datatype
DFD	2020-02-24T14:35:00	UTC Time (note missing Z)
KSAT	2020-02-24T14:35:00Z	UTC Time, explicit "Z" for "Zulu"

We can see from the provided examples, that as long the date formats are following ISO rules, the conversion between them is not much of the problem for todays systems, and in most programming, languages exist libraries doing this. Nevertheless, the mentioned flexibility of ISO standard does not obligate anybody to use any of its proposed formats for specific use. And so, for example the DFD interface uses the date format for UTC time without letter "Z". Most of the converters would however treat the date as local time and shift it to the UTC accordingly. In this case we've had to apply special treatment in our import and export routines, where we respectively add or remove the "Z" before the file leaves our system.

In case of legacy GSOC interfaces it gets even more complicated. The interface consists of 4 different representations, each having actually different date format. This was caused by organic growth of the system. When adding new format, the developers tried to improve the date format, but at the same time old formats could not be changed easily, because they were used in international context.

Please note the especially difficult GSOC Legacy Text Schedule format. It provides date and time in separate fields or columns of the file, which are just separated by spaces, and the date itself is also formatted with space separated values, and the year is coded as only two-digit value. For today's audience it may sound extremely strange, but people who know computer systems from the eighties will recognise the pattern. The main issue in those times was computer memory. It was sparse, so in any place where it was possible, programmers saved on characters, taking a lot of assumptions into account. Also, the files were often treated as character arrays, thus having number "20" in first two characters of the text line could be easily interpreted as year 2020. The file format was relying on specific array location of specific data in the file.

The following GSOC interfaces improve a bit, trying to provide date format in more clear way. Finally, the XML schedule format gets to actual ISO formatted date and time, however again with speciality of missing "Z", but adding the milliseconds, which in case of scheduling information for space communication does not make much sense.

Today such handling of data is very uncomfortable. It requires custom parsing of the file, with not very sophisticated code, which however still requires work. Tasks of parsing files are today covered by standard parsers (like for XML, JSON, ISO, etc.), they however cannot cope with such specific formatting.

Usage of different time date and time formats can easily lead to ambiguities. Already mentioned not using the explicit "Z" for UTC time can cause large time differences. Usage of milliseconds, which obviously can't be measured and used precisely, can lead to wrong identification of satellite contacts. Knowing all these differences beforehand helps of course to avoid big problems in operations, but it requires significant software development and testing in this area. Still there is a risk of having a special case of the date mismatch not covered by software, which can appear in some time. The best example of that are date conversion specialities related to year change (i.e. from December 31st to January 1st). This kind of trap is especially bad, as it hits the operative systems at the time, where agencies and companies are sparsely staffed due to holiday time.

4.2 Information Content

The information conveyed by scheduling processes is relatively simple for human operators. Spacecraft name, used ground antenna, start and stop times as well as some rudimentary, bilaterally agreed, configuration information is being exchanged. As soon we move it into the software based, automated processing scope, the apparent simple information starts to be ambiguous, complex and leading to misunderstandings.

Just simple reuse of information content, as previously used between human operators, causes in best case an avalanche of assumptions which programmers need to account for. For example, as presented in previous chapter the time format without "Z" requires the programmer to take that special treatment (described in respective ICD of the interface) into account. This works generally, but still provides a potential inconsistency between different interfaces in case of conversion.

When we talk about assumptions, one of the huge impacting factors are assumptions on the information which is not provided. In old scheduling processes, actually one could easily say, that most of the information required for the support was provided upfront in form of contract documentation, statement of work, ICD's or so-called pre-pass briefings. In the actual scheduling exchange, this information was not present, just assumed both partners know what they are doing. Now in case of fully automated software system, the question arises, does the system know all of this indirect information? And if not (which is mostly the case), how will the system know, which parameters shall have what values for specific support? Again, here the software engineers are asked to augment for missing explicit information by making assumptions in the system. Does the request include uplink? If yes, the software engineer needs to either try to collect the modulation, EIRP, polarisation information from some local configuration file or needs to select some default, kind of "one-fits-all" variant. In few cases it may work well, but many times it won't be in the sense of the requestor.

As with the missing information, there is also an issue with excess one. It's not as bad, but still requires piece of software augmentation to handle the interface in smooth way. The most popular example of such additional information is a comment. Sometimes it is provided by requestor, and the question here is, does it have any specific meaning for the automated system. If yes, then does it need to be parsed and recognised somehow? Is it some viable information? Maybe in that case it should not be provided as a comment? Another example is additional configuration, which may be used for other antennas in the network of specific mission, and was provided to our system only to keep interfaces unified. In that case, our system needs to roughly recognise and ignore such excess data, while allowing processing of the rest of the request at the same time (and not just crash due to mismatch in the format). Special case of this happens, if our system plays a role of gateway to further providers, which, contrary to us, can process or even require the excess information in the request. In that case our system needs not only to smoothly ignore the information, but at the same time keep the content intact, and forward it to third party providers.

References to ancillary information are another field which poses some potential problems. Most of the interfaces do not use references at all (which in case of for example TLE or generally pointing data leads actually to the absence issue discussed before). The ones that do, use some arbitrary paths to files located somewhere, which again does not guarantee all of the required information is really available. Another issue with references in general is, that their uniqueness needs to be guaranteed to avoid confusion and even database corruption. This leads to sophisticated mechanism, which has to cover for all of these potential issues and guarantee uniqueness of the reference and in effect correct execution of support.

One of the data mismatches, which we did not really expect to impact our developments in large scale, was the naming of spacecraft, ground stations and antennas. To some extent, this is a specific case of previously described referencing issue, if we assume the name of spacecraft or antenna is a reference to it. Typically used in space environment are short acronyms like WHM for Weilheim Ground Station, SGS for Svalbard Ground Station, TSX for TerraSAR-X spacecraft, etc. Using older ways of scheduling, sometimes we encountered small misunderstandings, mostly however it worked pretty well, and we assumed it as given, that everybody uses the same acronyms. At the moment we started to work with machine-to-machine interfaces, eventually we figured out, that every single third-party interface uses either completely different naming or individual items are named different. The differences are often not big, sometimes just one single letter in acronym is different, or acronym is made out of 4 and not 3 characters. Something pretty obvious for humans, poses however issue for software systems – different acronym is just different, the system cannot decide to arbitrary assign external acronym to the internal one (we restrain here from some sophisticated machine-learning or neural-network based software).

The solution we eventually had to implement is to introduce relatively wide stretched aliasing system, which allows us for each single interface to search and replace (or match) specified acronyms. That way the integrity and clarity of our database is guaranteed. Of course, as soon the information leaves our system again in the direction of external partner, the acronym matching takes place again, to present the information in the form known to this partner. There are some efforts from CCSDS to cope with that in form of centralised registry of spacecraft as well as sites and antennas, located in SANA, however especially the latter one is still not completely finished, and they are being until now kind of ignored by the community. With advent of new standards explicitly using them (like CSSM) it will hopefully change.

Further issue involves handling of units of measurable values. Typically, they will be provided just as a number, correlated to specific parameter name, for example "radiated_power = 100". Unless agreed somehow else, this poses serious issue. Power can be expressed in watts or decibels, and also milliwatts or kilowatts can be used. Taking our example, we cannot actually say what unit is being used. Within a single interface it can be somehow agreed, as soon we however perform interface matching, we have to carefully look at potentially different units and perform respective conversion where needed.

4.3 Service Package Status

The largest part of our interface matching activities is focusing on information state. We were expecting that some of the states can have different naming, but what we encountered was actually different understanding of the state machine for different information types. Also exchange of the messages over simple FTP imposes side effects on how the states need to be handled.

In our development we decided from the very beginning to follow the preliminary state machine model as planned for CSSM, especially centered around Service Package state machine (see Fig 16). After an operation which leads to creation of the service package (Create Service Package – CSP), the Service Package (SP) is being instantiated in our database, and gets its initial state of "Created". From here, depending if it was rejected by provider (for example due to conflict) or deleted by the user (for example due to no need anymore), the SP gets one of the final states, "Rejected" or "Cancelled" respectively. Otherwise, SP will get status "Scheduled" as soon it was decided by provider to firmly reserve resources for specific support. When time of the support comes, the state changes to "Executing", and when finished, ends up with a final state of "Archived". In case due to whatever failure the support has been broken, the final state gets value of "Aborted" to indicate for later accounting not successful support.

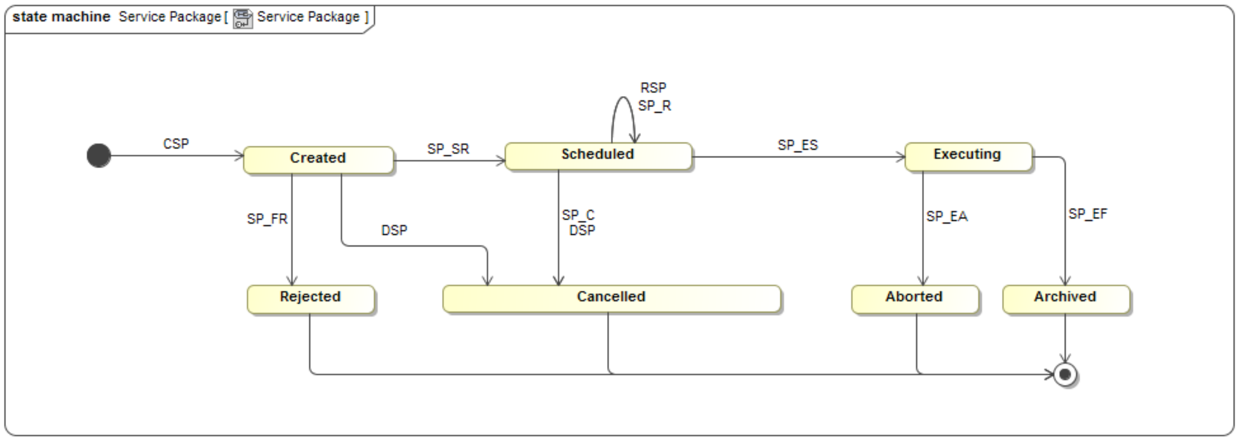

Fig. 16 CCSDS Service Management - preliminary Service Package State Machine (the acronyms on transitions represent operations or notifications which trigger or occur during transition).

The state machine of DFD is quite different, as presented on Fig. 17. There is no initial explicit state, thus the transitions to "Planned", "Scheduled", "Conflict" or first decision gate A can happen immediately. The "Planned" state has a specific meaning, as it could be located somewhere between ours "Created" and "Scheduled". It shows, that the support has been requested but not yet confirmed. From here the first decision A takes place, where in planning process the check on conflicts is made, and depending on the result, the support gets status "Scheduled" or "Conflict", whereas the latter is also a final state. The "Scheduled" state corresponds more or less to our "Scheduled" state, whereas "Approved" state is something between "Scheduled" and "Executing" in our state machine and persists over the support execution.

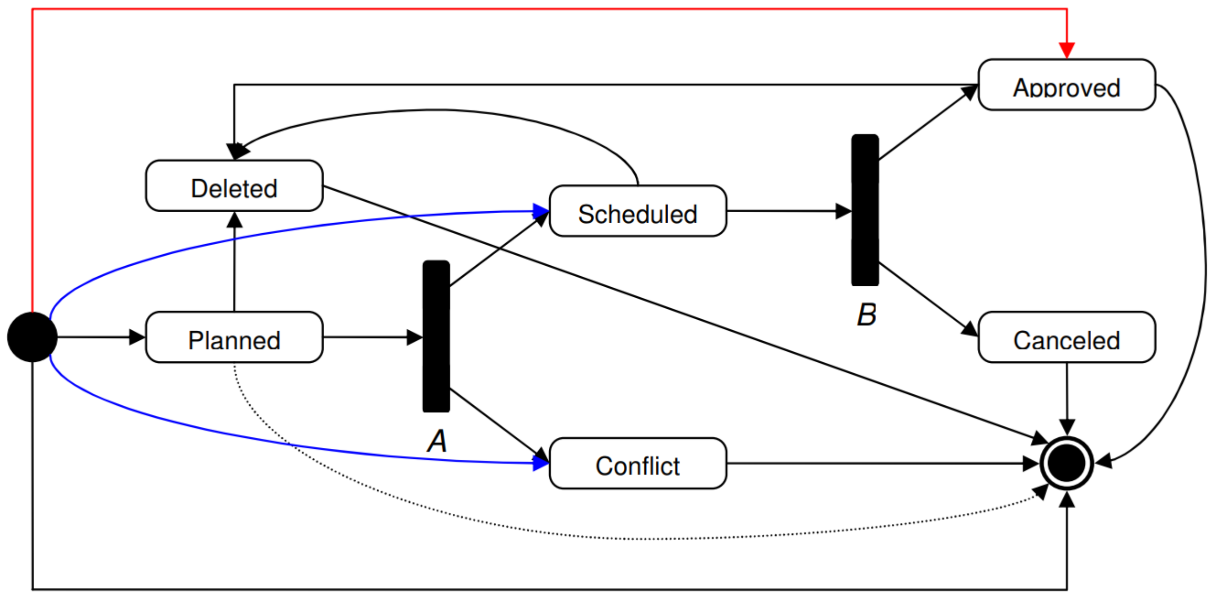

Fig. 17 DFD Scheduling State Machine

Table 3 shows the initial status of our analysis between CCSDS CSSM SP states and the DFD ones. For simplicity, we omit states which do not play a role here. The table shows in the first column the initial service package state at GSSNG, the first row shows the GSSNG state after exchange with external partner, and the table content shows the states provided by the external partner.

Table 3 State transitions between CCSDS and DFD I/F - initial status (Green – GSSNG states, Red DFD responses)

From\To	Created	Rejected	Canceled	Scheduled
Created	PLANNED	CONFLICT	**NA**	SCHEDULED
Scheduled	**NA**	**NA**	CANCELLED / DELETED	SCHEDULED / APPROVED

Unfortunately, such state matching did not work well in our first tries. As we perform the file exchange over FTP, there is an inherent delay in information and no immediate feedback is possible. This led very quickly to inconsistencies in the database, especially if the user in our system tried to cancel previously requested support (without waiting for any feedback). This led to an effect, that the support has been deleted already in our system, but after some time there arrived a confirmation to original request. Suddenly we have had a confirmed support from external provider in our system, which was not matching to anything locally anymore.

We decided therefore to introduce additional transactional states, which are not clearly visible to the users, but are needed internally in the database to cope with intrinsic delays of the operations and avoid issues as mentioned.

Table 4 CCSDS to DFD state matching table after implementation (Green – GSSNG states, Red – DFD states, Orange – transactional states, NA – state transition not applicable)

From\To	Rejected	Canceled	Scheduled	Created + Deletion pending	Created + Request pending	Scheduled + Deletion pending
Created	**NA**	**NA**	**NA**	Delete request sent to DFD	Schedule request sent to DFD	**NA**
Created + Deletion pending	**NA**	CANCELLED/ DELETED	SCHEDULED	**NA**	**NA**	**NA**
Created + Request pending	CONFLICT	**NA**	SCHEDULED	**NA**	PLANNED	**NA**
Scheduled + Deletion pending	**NA**	DELETED	**NA**	**NA**	**NA**	SCHEDULED
Scheduled	**NA**	CANCELLED	APPROVED	**NA**	**NA**	Delete request sent to DFD

Adding a meta-state (i.e. an overlapping state) is necessary to cope with delay due to ingestion of deletion/schedule request by the external system, as well as coping with failed message exchange due to networking problems, unavailability of service, etc. So, it can be said that, while the first state reflects on the current condition of the service package itself (i.e. answers to the question: is it scheduled to be executed or not?) the second state reflects on the result of the communication between service user and provider (i.e. answers to the question: was the information delivered?). The transactional states inhibit further state transition, making it dependent on actual feedback from external interface.

These transactional states helped us to keep database in sync with external partners. While testing, we discovered another side issue: our users complained, that the cancellation of passes for DFD station did not work immediately, and they had to perform the operation several times. It showed up that the human factor plays a role – people using the web-based GUI were used to relatively immediate system response. In case of DFD interface the transactional states (remember – not visible to users!) were kind of delaying the execution. And this annoyed our testers. We decided to show the transactional states additionally, so after deleting the already scheduled support, the user gets the state presented as "Scheduled / Deletion request pending". This finally solved the issue and pacified our testers.

In case of the KSAT interface, we encountered a very similar issue (and as already mentioned, literally every interface we had touched on, has it as well). The KSAT interface knows basic two states which are commonly communicated: "ACCEPTED" and "REJECTED". There is actually another one, "CANCELLED", however this one is used only explicitly in a case where the already booked pass is being cancelled due to the provider action (i.e. antenna unavailability due to failure). In other words, all of the interaction between customer and the provider is using two states only. This required us to implement multiple information handling mechanisms on that interface, to accommodate crossover to our database and our, CCSDS-like, states.

Table 5 shows our final solution on KSAT interface. As one can see, we actually managed to have similar handling as on DFD interface and respective transactional states are identical. Please note specific behaviour of KSAT interface in reaction to user initiated deletion requests. It does not confirm the deletion explicitly, but rather the absence of the previously requested support is to be understood as such. This requires from our software to actually parse complete schedule information and compare it with the previous one, to asses on absence of specific supports.

Another interesting observation from our analysis was that human users managed to revive already cancelled or deleted supports (if needed) to simplify their work flow. Such transition (from Cancelled or Deleted state back to Scheduled) is not foreseen by software systems, even theoretically possible. Once the support has been marked as Deleted or Cancelled it is often removed from the database altogether, thus simple instantiation is not possible, and more sophisticated mechanisms would need to be employed.

Table 5 CCSDS to KSAT state matching after implementation (Green – GSSNG states, Red – KSAT states, Orange – transactional states, NA – not applicable)

From\To	Rejected	Cancelled	Scheduled	Created + Deletion pending	Created + Request pending	Scheduled + Deletion pending
Created	**NA**	**NA**	**NA**	Delete request sent to KSAT	Schedule request sent to KSAT	**NA**
Created + Deletion pending	**NA**	Support not available in schedule anymore.	ACCEPTED	**NA**	**NA**	**NA**
Created + Request pending	REJECTED	**NA**	ACCEPTED	**NA**	**NA**	**NA**
Scheduled + Deletion pending	**NA**	Support not available in schedule anymore.	ACCEPTED	**NA**	**NA**	**NA**
Scheduled	**NA**	CANCELLED	**NA**	**NA**	**NA**	Delete request sent to KSAT

6. Conclusions

The analysis and implementation efforts presented in this paper, show that allegedly simple interfaces can impose large software development implications, including very complex algorithms, for solving the mismatching of these interfaces. Already coping with different file formats, especially the ones which were plain text based, creates overhead on additional code, which needs to be fully custom programmed. The information content is potentially not directly comparable and needs additional processing to avoid misunderstandings and, ultimately, bad operational support. The actual state machines of the information are partially very different between organisations, which poses not only issues on the interface matching, but potentially can impact actual operational execution.

The presented examples are shown not in order of the best solution, nor to bash on older interfaces, but rather to emphasize, that in existing operational environments it is not easy to implement station booking system without a need to implement (at least temporarily) old legacy interfaces. And that implementation causes considerable efforts in architecture, processes and programming areas. Our work shows the efforts connected to that and actual reason for the delay in complete system implementation is due to this.

Many interfaces which are coming into life currently (like new KSAT OGS interface) are much better than the ones presented here, and especially the formats (because of wide usage of JSON or XML) or transitional states (because of direct and immediate communication over REST) are not an issue anymore. Nevertheless, missing information content or misunderstandings in service or support life cycle are still present. This hopefully will change with a usage of already existing and upcoming CCSDS based Service Management standards, whereas convincing everybody to use it may still be a challenge, and obvious advantages rise first in real multi-partner cross support environment. As long any scheduling system is used only bilaterally, reimplementation to new CCSDS standard will stay on low priority list.

Acknowledgements

We would like to give our sincerely thanks to our colleagues from DFD and KSAT for allowing mentioning their interfaces in scope of our paper, as well as providing us with respective documentation.

References

[1] CCSDS 901.1-M-1, Space Communications Cross Support — Architecture Requirements Document, Magenta Book, Washington, May 2015
[2] CCSDS 910.4-B-2, Cross Support Reference Model—Part 1: Space Link Extension Services, Blue Book, Washington,
[3] Starlink - https://www.starlink.com/, accessed 25.01.2023
[4] CCSDS 911.1-B-4, Space Link Extension--Return All Frames Service Specification, Blue Book, Washington,
[5] CCSDS 911.2-B-3, Space Link Extension--Return Channel Frames Service Specification, Blue Book, Washington,

[6] CCSDS 910.11-B-1-S, Space Communication Cross Support—Service Management—Service Specification, Blue Book, Washington,
[7] CCSDS 902.0-G-1, Extensible Space Communication Cross Support--Service Management—Concept, Green Book, Washington
[8] Cross Support Service Management—Simple Schedule Format Specification. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.1-B-1. Washington, D.C.: CCSDS, May 2018
[9] Cross Support Service Management—Communications Planning Information Format. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.2-B-1. Washington, D.C.: CCSDS, March 2022
[10] Service Management Utilization Request Formats. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.9-B-1. Proposed.
[11] Cross Support Service Management—Service Package Data Formats. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.4-B-1. Proposed.
[12] Cross Support Service Management—Service Agreement and Configuration Profile Formats. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.5-B-1. Proposed.
[13] Cross Support Service Management—Space Link Event Sequence Data Format. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.6-B-1. Proposed.
[14] Cross Support Service Management—Service Accounting. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.8-B-1. Proposed.
[15] Cross Support Service Management—Management Services. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.10-B-1. Proposed.
[16] SANA - https://sanaregistry.org/, accessed 25.01.2023
[17] M. Gnat at al, Scheduling as an interoperability service and its security aspects, SpaceOps 2014, Pasadena, USA
[18] KSAT Scheduling and Post Pass Reporting ICD, 09.02.201, 10-10101-A-Doc, Issue 1.0, Kongsberg Satellite Services AS
[19] KSAT Acquisition availability – XML format description for customers, 05.05.2011, 10-10104-A-Doc, Issue 1.3.1, Kongsberg Satellite Services AS
[20] GSN Scheduling Interface Control Document, 2012-DLR-DFD-GSN-0007, Version: 1.05 23.02.2012, DLR e.V.
[21] ISO 8601 - https://www.iso.org/iso-8601-date-and-time-format.html, accessed 25.01.2023
[22] KSATLite https://www.ksat.no/ground-network-services/ksatlite/, accessed 25.01.2023
[23] LeafSpace https://leaf.space/, accessed 25.01.2023
[24] SSC Satellite Ground Stations https://sscspace.com/services/satellite-ground-stations/ , accessed 25.01.2023
[25] AWS Ground Station Service https://aws.amazon.com/ground-station/, accessed 25.01.2023