Interface definitions

Domino-X Earth Observation Project

Exported on 2023-11-16 16:57:13

Table of Contents

1 DOMINO-X interfaces 6

1.1 Alarms 6

1.1.1 Introduction 6

1.1.2 Dynamics of exchanges 6

1.1.3 Flow details 6

1.2 Booked Contacts 8

1.2.1 Introduction 8

1.2.2 Dynamics of exchanges 9

1.2.3 Request/Response 9

1.2.4 Data type 10

1.3 Booking Availability 10

1.3.1 Introduction 10

1.3.2 Dynamics of exchanges 10

1.3.3 Request/Response 11

1.3.4 Data type 12

1.4 Booking Cancellation 12

1.4.1 Introduction 12

1.4.2 Dynamics of exchanges 12

1.4.3 Request/Response 13

1.4.4 Data type 13

1.5 Booking Request 13

1.5.1 Introduction 13

1.5.2 Dynamics of exchanges 13

1.5.3 Request/Response 14

1.5.4 Data type 14

1.6 Contact Preparation 15

1.6.1 Introduction 15

1.6.2 Dynamics of exchanges 15

1.6.3 Request/Response 15

1.6.4 Data type 16

1.7 Contact Request 16

1.7.1 Introduction 16

1.7.2 Dynamics of exchanges 16

1.7.3 Request/Response 17

1.7.4 Data type 18

1.8 Coverage ProgR Feasibility Request 18

1.8.1 Introduction 18

1.8.2 Dynamics of exchanges 18

1.8.3 Request/Response 19

1.8.4 Data type 19

1.9 Coverage ProgR Manual Reassessment 20

1.9.1 Introduction 20

1.9.2 Dynamics of exchanges 20

1.9.3 Request/Response 21

1.9.4 Data type 21

1.10 Donwload & Upload Feasibility 22

1.10.1 Introduction 22

1.10.2 Dynamics of exchanges 22

1.10.3 Request/Response 22

1.10.4 Data type 23

1.11 Health Status 24

1.11.1 Introduction 24

1.11.2 Dynamics of exchanges 24

1.11.3 Flow details 24

1.12 Integrity and Traceability Records 26

1.12.1 Introduction 26

1.12.2 Dynamics of exchanges 26

1.12.3 Request/Response 26

1.12.4 Data type 27

1.13 Maneuver Plan 27

1.13.1 Introduction 27

1.13.2 Dynamics of exchanges 28

1.13.3 Request/Response 28

1.13.4 Data types 29

1.13.5 Example 29

1.14 Manoeuver Slots 30

1.14.1 Introduction 30

1.14.2 Dynamics of exchanges 30

1.14.3 Request/Response 31

1.14.4 Data type 31

1.15 Mission Contraints 32

1.15.1 Introduction 32

1.15.2 Dynamics of exchanges 32

1.15.3 Request/Response 32

1.15.4 Data type 33

1.16 Mission Plan 33

1.16.1 Introduction 33

1.16.2 Dynamics of exchanges 34

1.16.3 Request/Response 34

1.16.4 Data types 35

1.16.5 Example 36

1.17 Mission Reprogramming Request 36

1.17.1 Introduction 36

1.17.2 Dynamics of exchanges 36

1.17.3 Request/Response 37

1.17.4 Data type 38

1.18 Observable 38

1.18.1 Introduction 38

1.18.2 Dynamics of exchanges 38

1.18.3 Flow details 39

1.19 On-board report 40

1.19.1 Introduction 40

1.19.2 Dynamics of exchanges 40

1.19.3 Request/Response 41

1.19.4 Data types 42

1.19.5 Example 42

1.20 Orbit Events 44

1.20.1 Introduction 44

1.20.2 Dynamics of exchanges 44

1.20.3 Request/Response 45

1.20.4 Data types 45

1.20.5 Example 45

1.21 Payload Calibration 46

1.21.1 Introduction 46

1.21.2 Dynamics of exchanges 46

1.21.3 Request/Response 47

1.21.4 Data type 48

1.22 Polarisation Plan 48

1.22.1 Introduction 48

1.22.2 Dynamics of exchanges 49

1.22.3 Request/Response 49

1.22.4 Data type 51

1.23 Predicted Orbits 51

1.23.1 Introduction 51

1.23.2 Dynamics of exchanges 51

1.23.3 Request/Response 52

1.23.4 Data type 53

1.24 Product Availability Notification to be consolidated (pick-up-point-light) 54

1.24.1 Introduction 54

1.24.2 Dynamics of exchanges 54

1.24.3 Request/Response 54

1.24.4 Data type 55

1.25 Product Integrity Verification Request 55

1.25.1 Introduction 55

1.25.2 Dynamics of exchanges 55

1.25.3 Request/Response 56

1.25.4 Data type 56

1.26 Production Context 56

1.26.1 Introduction 56

1.26.2 Dynamics of exchanges 57

1.26.3 Request/Response 57

1.26.4 Data type 58

1.27 Products to sign 58

1.28 Product Traceability Request 58

1.28.1 Introduction 58

1.28.2 Dynamics of exchanges 58

1.28.3 Request/Response 59

1.28.4 Data type 59

1.29 ProgR Activation Request 60

1.29.1 Introduction 60

1.29.2 Dynamics of exchanges 60

1.29.3 Request/Response 60

1.29.4 Data type 61

1.30 ProgR Analysis Request 61

1.30.1 Introduction 61

1.30.2 Dynamics of exchanges 61

1.30.3 Request/Response 62

1.30.4 Data type 62

1.31 ProgR Cancel Request 63

1.31.1 Introduction 63

1.31.2 Dynamics of exchanges 63

1.31.3 Request/Response 63

1.31.4 Data type 64

1.32 ProgR Consultation Request 64

1.32.1 Introduction 64

1.32.2 Dynamics of exchanges 64

1.32.3 Request/Response 64

1.32.4 Data type 65

1.33 ProgR Update Request 65

1.33.1 Introduction 65

1.33.2 Dynamics of exchanges 65

1.33.3 Request/Response 66

1.33.4 Data type 66

1.34 Routine Pivot Point Slots 67

1.34.1 Introduction 67

1.34.2 Dynamics of exchanges 67

1.34.3 Request/Response 67

1.34.4 Data type 68

1.35 Signed products 68

1.36 Site Monitoring Allocation Projection Request 68

1.36.1 Introduction 68

1.36.2 Dynamics of exchanges 68

1.36.3 Request/Response 69

1.36.4 Data type 70

1.37 System and Satellite Configuration 70

1.37.1 Introduction 70

1.37.2 Dynamics of exchanges 71

1.37.3 Request/Response 71

1.37.4 Data type 72

1.38 TM Values 73

1.38.1 Introduction 73

1.38.2 Dynamics of exchanges 73

1.38.3 Request/Response 73

1.38.4 Data types 75

1.38.5 Example 75

1.39 Unavailability Slots 76

1.39.1 Introduction 76

1.39.2 Dynamics of exchanges 76

1.39.3 Request/Response 77

1.39.4 Data type 78

1.40 User Request Activation Request 78

1.40.1 Introduction 78

1.40.2 Dynamics of exchanges 78

1.40.3 Request/Response 79

1.40.4 Data type 80

1.41 User Request Consultation Request 81

1.41.1 Introduction 81

1.41.2 Dynamics of exchanges 82

1.41.3 Request/Response 82

1.41.4 Data type 83

1.42 User Request Detailed Consultation Request 83

1.42.1 Introduction 83

1.42.2 Dynamics of exchanges 84

1.42.3 Request/Response 84

1.42.4 Data type 85

1.43 User Request Update Request 85

1.43.1 Introduction 85

1.43.2 Dynamics of exchanges 85

1.43.3 Request/Response 86

2 Standard interfaces 87

2.1 CCSDS TC SPACE DATA LINK PROTOCOL (232.0-B-3) 87

2.1.1 Introduction 87

2.1.2 Applicable documents 87

2.1.3 DOMINO-X logical interfaces 87

2.2 CCSDS TM SPACE DATA LINK PROTOCOL (132.0-B-2) 87

2.2.1 Introduction 87

2.2.2 Applicable documents 87

2.2.3 DOMINO-X logical interfaces 87

2.3 ESA-AUXIP V1.2 interfaces 88

2.3.1 Introduction 88

2.3.2 Applicable documents 88

2.3.3 DOMINO-X logical interfaces 88

2.3.4 Climato 88

2.3.5 Data Elevation Model 89

2.3.6 GIPP 89

2.3.7 Ground Reference Images 89

2.3.8 IERS 89

2.3.9 Weather Forecasts 89

2.4 ESA-LTAIP V1.7 interfaces 90

2.4.1 Introduction 90

2.4.2 Applicable documents 90

2.4.3 DOMINO-X logical interfaces 90

2.5 ESA-PRIP V1.7 interfaces 90

2.5.1 Introduction 90

2.5.2 Applicable documents 90

2.5.3 DOMINO-X logical interfaces 90

2.6 ESA-XBIP V1.1 interfaces 91

2.6.1 Introduction 91

2.6.2 Applicable documents 91

2.6.3 DOMINO-X logical interfaces 91

2.7 OGC API Processes V1.0.0 interfaces 91

2.7.1 Introduction 91

2.7.2 Applicable documents 92

2.7.3 DOMINO-X logical interfaces 92

2.7.4 Falsification Detection Request & follow-up 93

2.7.5 Image Signature Request & follow-up 93

2.7.6 PPS reprocessing request & follow-up 93

2.8 STAC API V1.0.0 interfaces 93

2.8.1 Introduction 93

2.8.2 Applicable documents 93

2.8.3 DOMINO-X logical interfaces 93

2.8.4 Catalog Structured Search 94

2.8.5 Product Metadata Update 94

2.8.6 Product Retrieval 95

# DOMINO-X interfaces

## Alarms

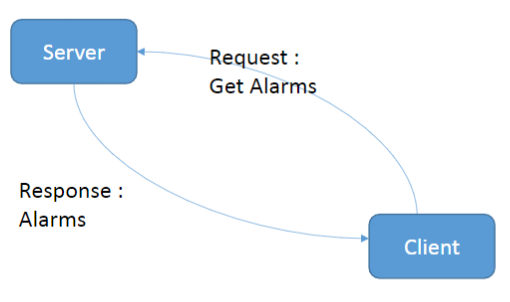
### Introduction

Each domino can raise an alarm in order to inform the operator of an abnormal behaviour. These alarms are gathered by the OMS via the get alarms transaction.

Each alarm remains under its original domino responsibility, which is in charge to monitor that the alarm remains relevant and to store the alarm history.

### Dynamics of exchanges

Periodically, the OMS sends an HTTPS/REST "**GET ALARMS**" request to the dominoes. In response, the requested domino computes and returns the requested monitoring information.



[Alarms.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641777/Alarms.pptx?api=v2&modificationDate=1680252159000&version=1)

These Web Service are based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Flow details

#### Request

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **request** |
| GET **Alarms**request | Ask for current alarms list of targeted domino | * **start date**: starting date of queried time slot |

#### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | **Code Status** | **Description** | **JSON response** |
| **Alarms**response | 200 | List of alarms that are (have been) active in the requested time period. That concerns   * Alarm still active, * Alarm raised before the start date and closed during the period, * Alarm raised and closed during the period. | A list of alarms. Each alarm is characterized by the following fields:   * + **Criticality** represents the importance of the alarm.   + **Type** used in order to display it in the proper section of the HMI and to the proper operator   + **Date** represent the date/time of the alarm creation   + **Resolution Date** represent the date/time of the alarm disappearance     - Present only when the alarm disappear   + **UID**: Alarm instance unique ID     - Uniqueness at domino level     - Allowing to identify the same alarm in different query   + **Short Description**: A short description of the alarm     - Can be used as a title   + **Long Description**: Textual description of the alarm with customizable fields such as metric values |

*Note on fields values*

Except for Resolution Date, which is set later, all fields of an alarm remain constant during the whole life of the alarm.

#### Data types

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Type | Values | Description |
| Criticality | Enumerate | Critical | The main functions are not available  *Example: [MPS] domino lost access to its request database* |
| Serious | Major outage: A part of the services is not available  *Example: [FOS] “Extract GNSS data from TM” function is not working* |
| Caution | Partial outage: Minor anomalies, without severe impact  *Examples: Disk usage is over 80% ; Degraded performance* |
| Normal | All services are fully functional. |
| Standby | Transitory state, but nothing abnormal  *Examples: Starting phase or planned maintenance* |
| Date & Resolution Date | MM/DD/YY HH:MM:SS |  | Date |
| Long Description | String |  |  |
| Short Description | String |  | max size: **TBD** |
| Type | Enumerate  Note: A domino is not limited to one alarm type. For example, all dominoes can raise I.T. alarm | Spacecraft | Alarm linked to the spacecraft health status |
| I.T. | Alarm linked to a low level status in the infrastructure (Domino, cloud, infrastructure, network, hardware) |
| Ground Station | Alarm related to a physical or to a "As A Service" station, to the baseband or to the booking capability, |
| Mission Programming | Alarm related to programming capacity of sovereign or external system in mono or multi mission (including the reprogramming capacity) |
| Data processing | Product processing starting form the TMI conversion to pivot format and ending with the cataloguing, archiving and distribution of the finished products. |
| Control Center | Spacecraft platform activities on ground (TC preparation, real time activities, HKTM processing) |
| UID | String |  | format: **TBD** |

## Booked Contacts

### Introduction

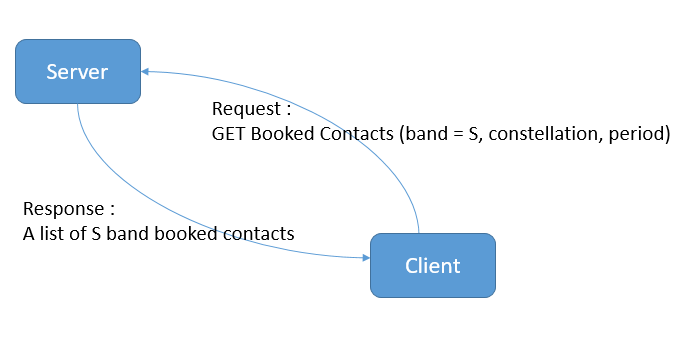
The "**Booked Contacts**" Interface allows clients to straightforwardly retrieve a list of booked contacts for a given band (X, S, etc.), constellation and period.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Booked\_Contacts.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307396817/Booked_Contacts.pptx?api=v2&modificationDate=1678714252000&version=1)

The client sends a HTTPS/REST "GET **Booked Contacts**" request with band, constellation and period as parameters. In response, in case of success, the server returns a list of booked contacts.



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| PPS, MPS, FOS | SCRMS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Booked Contacts** request | Ask for a list of booked contacts for a given band, constellation and period. | Query Parameters :     bands (X, S) = X     constellation     period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Booked Contacts** response | 200 | Successfully returned a list of booked contacts | List of booked contacts :  {     stationUri     stationPort     aos     los     constellation     satellite     band  }  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| aos | Date | Acquisition of signal (UTC time) |
| band | String | Name of the frequency band (as defined by ITU) |
| constellation | String | Satellite constellation name |
| los | Date | Loss of signal (UTC time) |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| stationPort | Integer | IP port of the station from where the data are received or sent. |
| stationUri | String | Station identifier |

## Booking Availability

### Introduction

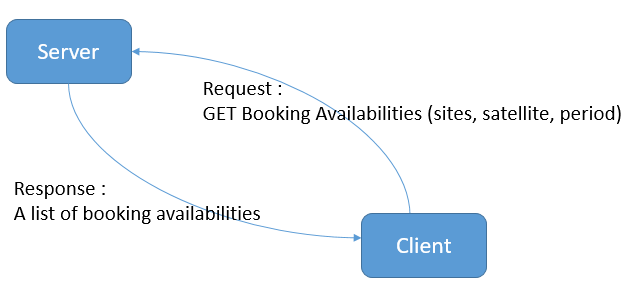
The "**Booking Availability**" Interface allows clients to straightforwardly retrieve a list of booking availabilities.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Booking\_Availability.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397885/Booking_Availability.pptx?api=v2&modificationDate=1678800335000&version=1)

The client sends a HTTPS/REST "GET **Booking Availability**" request with sites, satellite and period as parameters. In response, in case of success, the server returns a list of booking availabilities.



Clients/server example :

|  |  |
| --- | --- |
| Cients | Server |
| SCRMS | GSaaS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Booking Availability** request | Ask for a list of booking availabilities for a list of sites, a satellite and a period. | Query Parameters:     sites (list of sites)     satellite     period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Booking Availability** response | 200 | Successfully returned a list of booking availabilities | {     site     period     satellite     stations  }  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| sites | List of strings | List of geographical site names where the ground stations are located |
| stations | List of strings | List of station names (ground station names) |

## Booking Cancellation

### Introduction

The "**Booking Cancelation**" Interface allows clients to cancel a previous booked contact (with booking ID as request parameter)

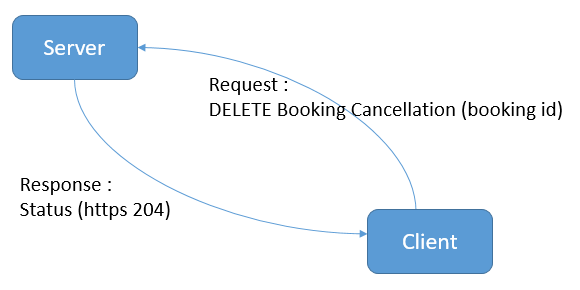
This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Booking\_Cancellation.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307398157/Booking_Cancellation.pptx?api=v2&modificationDate=1678805376000&version=1)

The client sends a HTTPS/REST "DELETE **Booking Cancellation**" with a booking ID as parameter. In response, the server returns a processing confirmation status of the request.

.



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| SCRMS | GSaaS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| DELETE **Booking Cancellation** request | Cancel a previous booked contact | {     booking ID  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| DELETE **Booking Cancellation** response | 204 | Successfully returned a status | status code 204 |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| booking ID | Integer | Booking Identifier |

## Booking Request

### Introduction

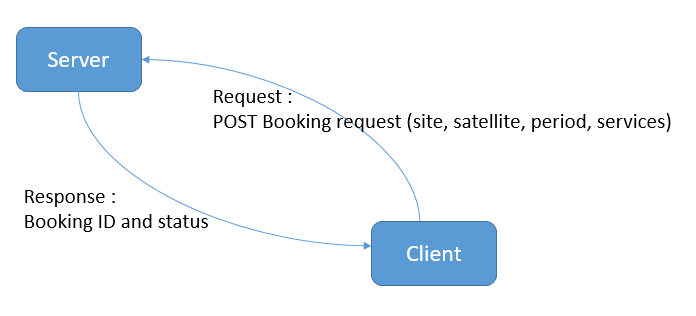
The "**Booking Request**" Interface allows clients to straightforwardly book frequency bands and link types (uplink/downlink) on a site, over a given period for a satellite.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Booking\_Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307398061/Booking_Request.pptx?api=v2&modificationDate=1678867955000&version=2)

The client sends a HTTPS/REST "POST **Booking Request**" with site , satellite, services and period as parameters. In response, the server returns a booking status and a booking ID (in case of success).



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| SCRMS | GSaaS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Booking Request** request | Request bands and link types on a site, over a given period for a satellite. | {     Site     Satellite     Period     Services [        {         band (S, X)         type (UPLINK, DOWNLINK)        }…     ]  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Booking Request** response | 200 | Successfully returned a booking status and a booking ID (in case of success a real booking ID) | {     Booking ID     Booking status (ACCEPTED, CANCELLED)  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| booking ID | Integer | Booking identifier |
| booking status | Enum of strings | Booking status ( ACCEPTED, CANCELLED) |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| services | Services | Structure type (band (S/X) and link type : UPLINK/DOWNLINK)) |
| site | String | Name of the geographical site where the ground stations are located |

## Contact Preparation

### Introduction

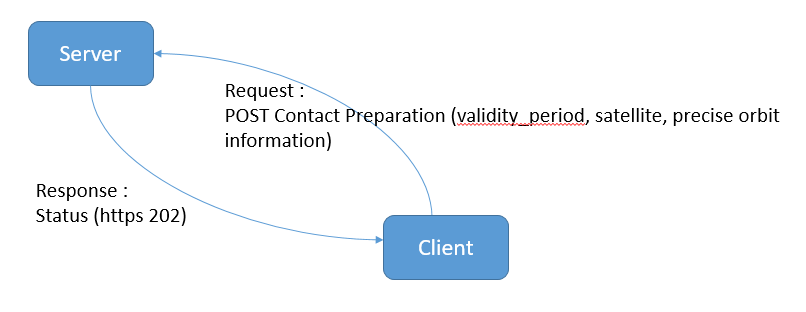
The **"Contact Preparation"** Interface allows clients to send contact preparation information to the server (such as orbit information).

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Booking\_Preparation.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307398181/Booking_Preparation.pptx?api=v2&modificationDate=1678806209000&version=1)

The client sends a HTTPS/REST "POST **Contact Preparation**" with a validity period, a satellite and a precise orbit information as parameters. In response, the server returns a processing confirmation status of the request.



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| SCRMS | GSaaS or Owned GS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Contact Preparation** request | Send contact preparation information | {  validity\_period  satellite  precise orbit information  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Contact Preparation** response | 202 | Successfully returned a status | status code 202 |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| precise orbit information | EphemeridList | List of ephemeris |
| satellite | String | Satellite name |
| validity period | Period | Start date, end date (validity time span of the transmitted data) |

## Contact Request

### Introduction

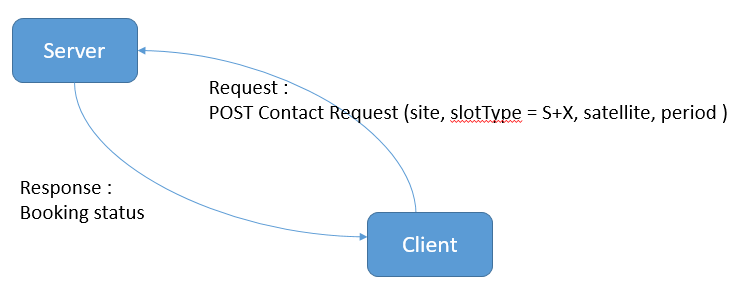
The *"***Contact Request***"* Interface allows clients to ask for adding a contact for a given slot type (ITM, TM/TC, both), constellation, satellite, site and time constraints.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Contact\_Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397445/Contact_Request.pptx?api=v2&modificationDate=1678978691000&version=1)

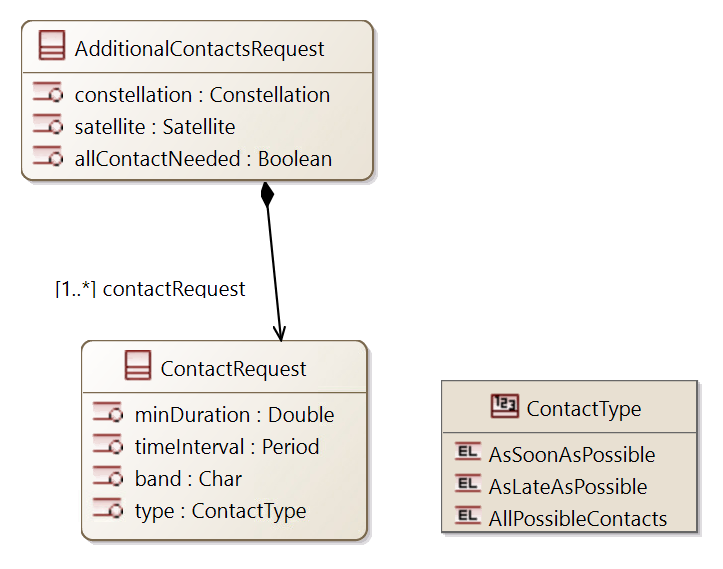
The client sends a HTTPS/REST "POST **Contacts Request**" request with slot type(ITM, TM/TC, both), satellite, site and period as parameters. In response, the server returns a booking status (OK, NOK).



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | SCRMS |

### Request/Response



|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Contacts Request** request | Ask to add a new additional contact | See Additional Contacts Request (diagram above) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Contacts Request** response | 200 | Successfully returned a booking status | {     status (OK, NOK)  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| deltaTime | Integer | Acceptable delta time from the targetTime |
| minDuration | Long | Minimum duration of the contact in seconds |
| satellite | String | Satellite name |
| slotType | String (Enum) | List of band names  (as defined by ITU) : S/X/S,X |
| targetTime | Date | Target Date (and time) for the contact to add |

## Coverage ProgR Feasibility Request

### Introduction

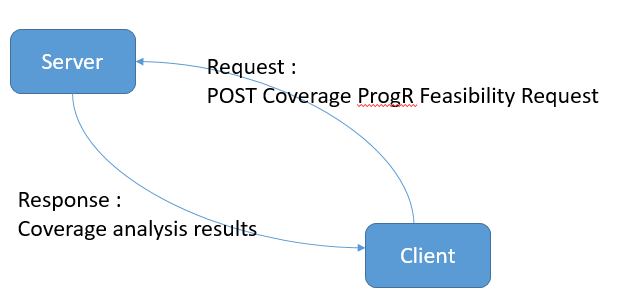
The "**Coverage ProgR Feasibility Request**" Interface allows a client to study the feasibility of a Coverage Programming Request (coverage ProgR). The client can ask to take the work load into account or not. This work load is a parameter of the request.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **Coverage ProgR Feasibility Request**" request with a programming request as parameter.

The exchange between the client and the server is synchronous.



[Coverage ProgR Feasibility Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907252/Coverage%20ProgR%20Feasibility%20Request.pptx?api=v2&modificationDate=1679677420000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Coverage ProgR Feasibility Request**request | Ask to study the feasibility of a Coverage Programming Request (coverage ProgR)  The client can ask to take the work load into account or not. This work load is a parameter of the request (use\_workload =  true/false). | {  programming request {     ...  (see [User Request Activation Request](#scroll-bookmark-54))  } |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Coverage ProgR Feasibility Request**response | 200 | In case of success return a coverage ProgR analysis results | {       constellation,       coverage data [         { date, progress [date, progresspercent), ...], is\_completed},         …       ]  },  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Constellation |
| coverage data | CoverageData | Progress status and progress history of the coverage data |
| programming request | ProgrammingRequest | Programming Request (see [User Request Activation Request](#scroll-bookmark-54)) |
| use workload | Boolean | To take into account (or not) the work load (false or true). |

## Coverage ProgR Manual Reassessment

### Introduction

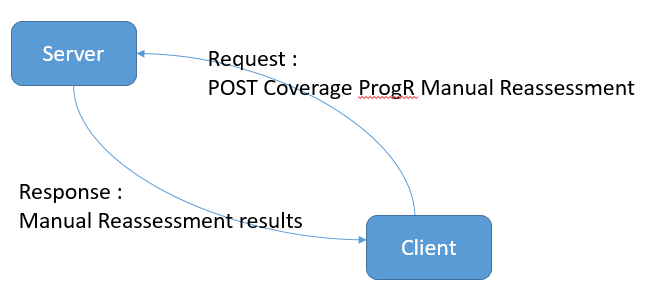
On operator demand, the "**Coverage ProgR Manual Reassessment**" Interface allows a client to study the feasibility of a Coverage Programming Request  (coverage ProgR Manual Reassessment). The client can ask to take the work load into account or not. This work load is a parameter of the request.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **Coverage ProgR Manual Reassessment**" request with a programming request (with its GUID) as parameter.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640695/Coverage%20ProgR%20Feasibility%20Request.pptx?api=v2&modificationDate=1679903129000&version=1)

[Coverage ProgR Manual Reassesment.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640695/Coverage%20ProgR%20Manual%20Reassesment.pptx?api=v2&modificationDate=1679903742000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Coverage ProgR Manual Reassessment**request | On operator demand, ask to study the feasibility of a Coverage Programming Request (coverage ProgR)  The client can ask to take the work load into account or not. This work load is a parameter of the request (use\_workload =  true/false). | {  programming request {     guid,     ...  (see [User Request Activation Request](#scroll-bookmark-54))  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Coverage ProgR Manual Reassessment** response | 200 | In case of success return a coverage ProgR analysis results | {       constellation,       coverage data [         { date, progress [date, progresspercent), ...], is\_completed},         …       ]  },  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Constellation |
| coverage data | CoverageData | Progress status and progress history of the coverage data |
| programming request | ProgrammingRequest | Programming Request (see [User Request Activation Request](#scroll-bookmark-54)) |
| use workload | Boolean | To take into account (or not) the work load  (false or true) |

## Donwload & Upload Feasibility

### Introduction

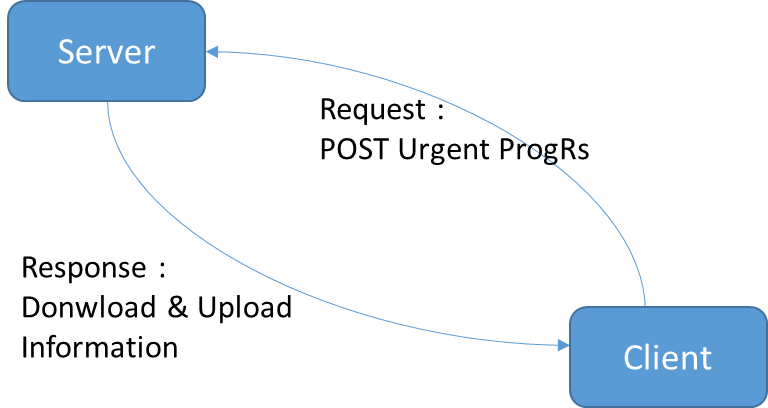
The "**Donwload & Upload Feasibility**" Interface allows clients to check the donwload and upload feasibility of several Programming Orders. This interface is used by [FS] to check from [MPS] that urgents requests could be fulfiled or need additional contacts.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

[Donwload\_upload\_feasibility.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/326369811/Donwload_upload_feasibility.pptx?api=v2&modificationDate=1686311386000&version=1)

The client sends a HTTPS/REST "POST **a list of ProgRs**" to be checked, the server returns for each Programming Request if there are sufficient contacts to upload and download it, if not specify potential additional contact that should be added.



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Donwload & Upload Feasibility** request | Check ] that urgents requests could be fulfiled or need additional contacts. | A list of Programming Request (see User Request model definition) |

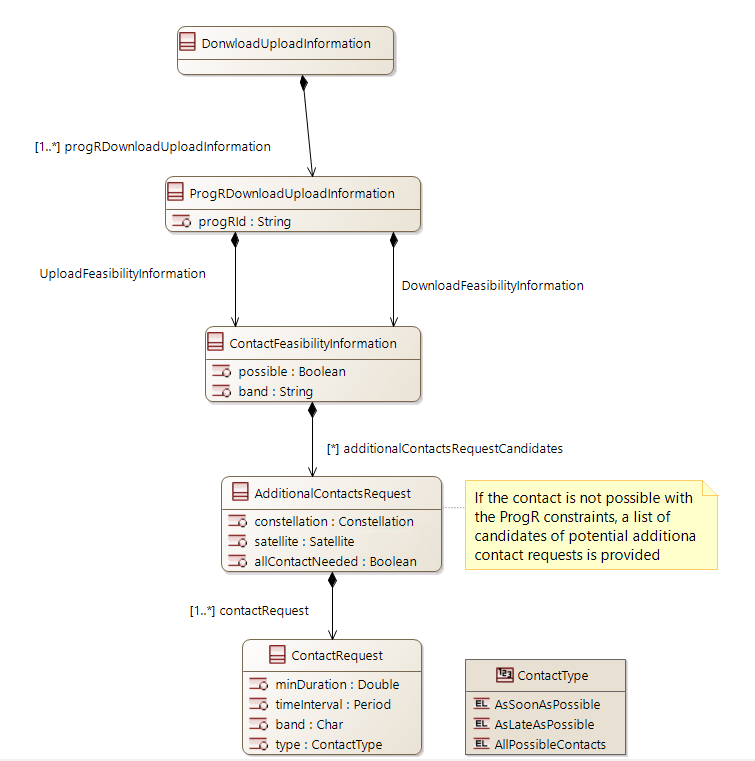
|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Donwload & Upload Feasibility** response | 200 | Successfully returned a booking status and a booking ID (in case of success a real booking ID) | DonwloadUploadInformation (see Data type bellow) |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

The following diagram describes the structure of the DownloadUploadInformation. Note that the AdditionalContactsRequest definition is also available into the Contact Request Interface.



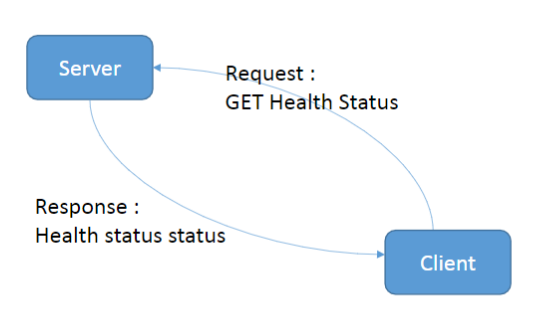
## Health Status

### Introduction

Health status allows the OMS to get a general synthesis of the targeted domino running status. In order to get a general knowledge of the whole platform status, the OMS query regularly all dominoes.

### Dynamics of exchanges

Periodically, the OMS sends an HTTPS/REST "**GET Heath Status**" request to the dominoes. In response, the requested domino computes and returns the requested monitoring information.  The OMS can request for the heath status outside the periodical queries, for instance on alarm reception with criticality different of currently  known heath status.



[HealthStatus.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641781/HealthStatus.pptx?api=v2&modificationDate=1680252206000&version=1)

These Web Service are based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Flow details

#### Request

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| **GET Health Status**request | Ask for a fresh Health Status of targeted domino | * **Historical Depth**: number of previous health status to return   *Optional (default value = 1)* |

#### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | **Code Status** | **Description** | **JSON response** |
| **Health Status**response | 200 | Current health status of the domino and previous states given to detects 'chattering' status | * **Health Status** represents the general status of the dominoes.   + Possible values are listed beside * **Status Duration** represent the time since entering the current state   + Computed on get request reception * **Previous Health Status** is the list of the previous dominoes status   + Expressed with status and date   + Number of entries depend on get parameter "Historical Depth" |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *404 Not Found*
* *500 Internal Server Error*

*A lack of response from the domino (error code 404 or500), to a health status request indicates a dysfunction of this domino or of the network*

#### Data type

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Type | Values | Description |
| Date | MM/DD/YY HH:MM:SS |  | Date |
| Health Status | Enumerate | Critical | The main functions are not available  *Example: [MPS] domino lost access to its request database* |
| Serious | Major outage: A part of the services is not available  *Example: [FOS] “Extract GNSS data from TM” function is not working* |
| Caution | Partial outage: Minor anomalies, without severe impact  *Examples: Disk usage is over 80% ; Degraded performance* |
| Normal | All services are fully functional. |
| Standby | Transitory state, but nothing abnormal  *Examples: Starting phase or planned maintenance* |
| Off | Domino fully stopped, or unresponsive, but nothing abnormal |
| Previous Health Status | List of status |  | A status is composed of   * Heath Status * Date: Entry date in this heath status   *Example with Historical Depth set to 4: {status: “Normal”, date: 17/06/2022 16h32}, {status: “Caution”, date: 17/06/2022 16h31}, {status: “Standby”, date: 11/06/2022 12h35}, {status: “”, date: }]* |
| Status Duration | Integer |  | Number of seconds |

## Integrity and Traceability Records

### Introduction

The "**Integrity and Traceability Records**" Interface allows clients (i.e production dominoes or any domino creating catalog entries) to straightforwardly push integrity and traceability information (i.e records) about their production to be centralized in DITS domino.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The following Dominoes use theIntegrity and traceability records API :

|  |  |
| --- | --- |
| Clients | Server |
| PPS | DITS |
| EPS | DITS |
| APS | DITS |
| DDS | DITS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| POST **Integrity and Traceability Records** push | Sends integrity and/or traceability info (hash codes, sources, providers, processors...) for a product ID |  |

|  |  |  |
| --- | --- | --- |
| **Interface** | **Status code** | **Description** |
| POST **Integrity and Traceability Records** response | 201 | Successful operation |
|  | 400 | Invalid or mal-formatted record |
|  | 409 | ProductID for the record already exists |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| **COMMON** |  |  |
| productId | String | PFD product ID |
| **INTEGRITY INFO** |  |  |
| productHash | String | Computed hash code |
| productHashType | String | MD5/SHA1/SHA256... |
| **TRACEABILITY INFO** |  |  |
| parentIDs | Array |  |
| productProvider | String |  |
| processorName | String |  |
| processorID | String |  |
| processorVersion | String |  |
| processorSteps | Array |  |
| processorParameters | Array |  |
| auxiliaryDataIDs | Array |  |

## Maneuver Plan

### Introduction

The "**Maneuver Plan**" Interface allows clients to retrieve the TC plan for satellite maneuvers.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requirements:

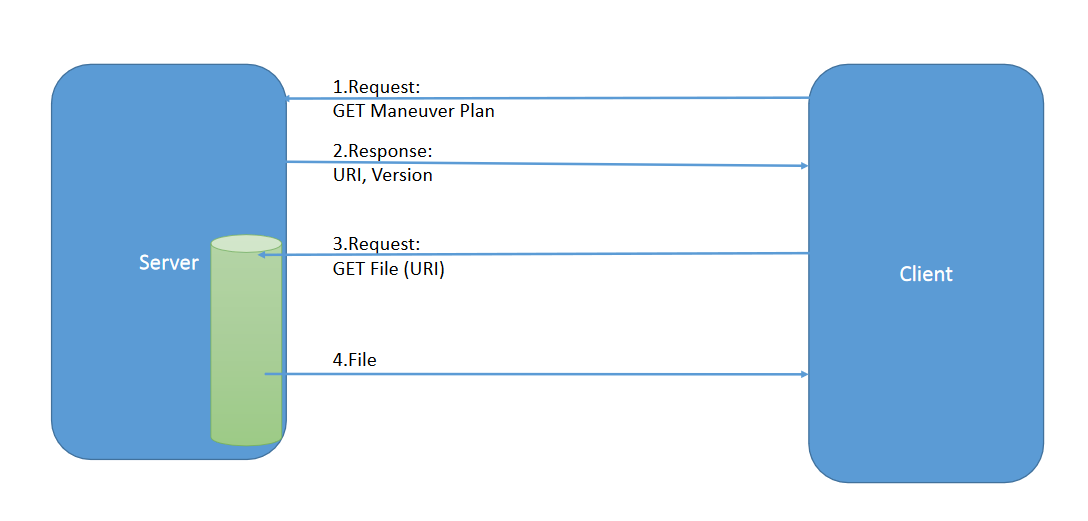
* Asynchronous Exchange;
* Centralized data;
* One pickup-point for each data producer;
* Data described in a file.

The interface format is JSON.

### Dynamics of exchanges

The exchange is a four steps process:

1. The client sends a HTTPS/REST "GET **Maneuver Plan**" request with selection parameters;
2. In response, the server returns the URI and version of thefile containing the requested plan;
3. The client checks the version against the version of its latest download. If needed, the client downloads the file;
4. In response, the server returns the requested file.



[maneuver\_plan.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/327975077/maneuver_plan.pptx?api=v2&modificationDate=1686665659000&version=1)

Typical client/server:

|  |  |
| --- | --- |
| Clients | Server |
| FOS | FDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **Maneuver Plan** request | Asks for a maneuver plan for a satellite and over a period | satellite,    period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **Maneuver Plan** response | 200 | in case of success, the URI and version of the file containing the maneuver plan | {    uri,    version  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **File** request | Requests the file for the given URI | URI |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **File** response | 200 | in case of success, returns the maneuver plan as a list of Time Tagged TC in an abstract language | {     metadata,    "Time-tagged TC":    [      "event": { eventClass, eventName, time, parameters }    ]  } |

*Note on status codes:*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data types

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| **PARAMETERS** |  |  |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| uri | String | File identifier |
| version | String | Version (or tag) of the file |
| **MANEUVER PLAN** |  |  |
| metadata | JSon object | metadata about the plan (satellite, horizon, production date...) |
| eventClass | String | a category for the event |
| eventName | String | the name of the event |
| time | Date | the planned execution time |
| parameters | JSon array | the parameters of the event |

### Example

Time-tagged TC example:

    {

      "eventClass": "thrustEvent",

      "eventName": "startThrust",

      "time": "2023-07-02T00:00:00.000",

      "parameters": [

        {

          "name": "level",

          "value": "low"

        }

      ]

    }

## Manoeuver Slots

### Introduction

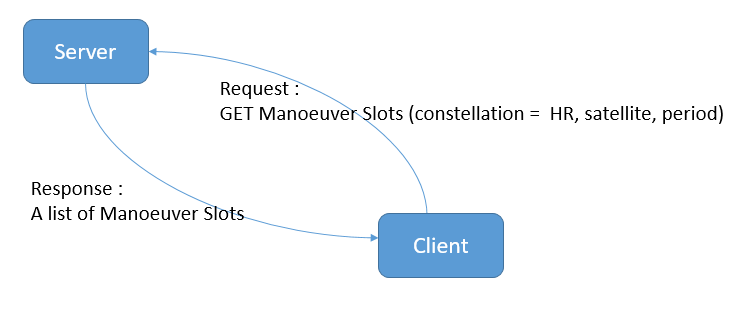
The "**Manoeuver Slots**" Interface allows clients to straightforwardly retrieve a list of manoeuver slots for a given constellation, satellite and period.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **Manoeuver Slots**" request with constellation, satellite and period as parameters. In response, in case of success the server returns a list of manoeuver slots over the period given as input to the request.

[Manoeuver\_Slots.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397138/Manoeuver_Slots.pptx?api=v2&modificationDate=1678717288000&version=1)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| MPS | SCRMS |
| SCRMS | FDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Manoeuver Slots** request | Ask for a list of manoeuver slots for a given constellation, satellite and period. | Query Parameters     constellation = HR     satellite     period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Manoeuver Slots** response | 200 | Successfully returned a list of manoeuver slots | List of manoeuver slots :  {     constellation     satellite     manoeuver period     type  }  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| manoeuver period | Period | Start date, end date (time slots of unavailability over the period given as input to the request) |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| type | String | Type of the manoeuver (CAM, OCM) |

## Mission Contraints

### Introduction

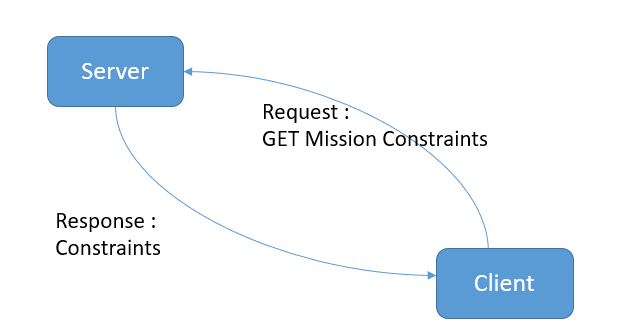
The "**Mission Constraints**" Interface allows clients to straightforwardly retrieve the mission programmation parameters and constraints for a constellation or a satellite.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **Mission Constraints**" request with constellation, satellite. In response, in case of success the server returns the Programming Parameters of a Programming Request and the constraints on these parameters (acceptable range for instance).

[Mission\_Constraints.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/347964136/Mission_Constraints.pptx?api=v2&modificationDate=1695113469000&version=1)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| GET **Mission Constraints** request | Ask for the Programming Parameters of a Programming Request and the constraints on these parameters (acceptable range for instance) | Query parameters :   * Constellation (optional, by default all) * Satellite (optional, by default all) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Mission Constraints**response | 200 | Successfully returned a list of misison constraints | List of mission constraints :  [     {        constellation,         [             {                   satellite,                   missionParamConstraints             }             ,…         ]      }      ,…  ] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| satellite | String | Satellite name |
| missionParamConstraints | object | Mission specific format  Corresponds to the Programming Parameters of a Programming Request and the constraints on these parameters (acceptable range for instance) |

## Mission Plan

### Introduction

The "**Mission Plan**" Interface allows clients to retrieve the TC plan for image acquisitions.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requirements:

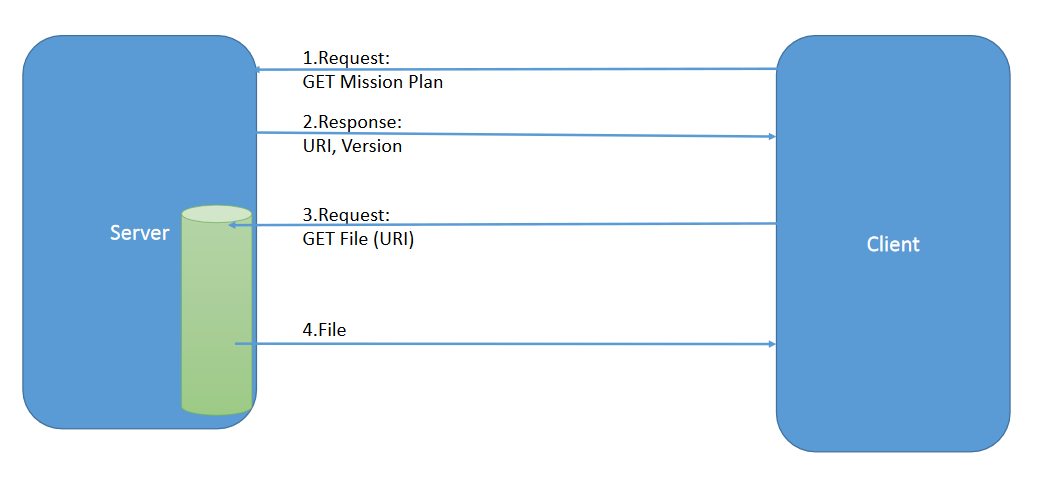
* Asynchronous Exchange;
* Centralized data;
* One pickup-point for each data producer;
* Data described in a file.

The interface format is JSON.

### Dynamics of exchanges

The exchange is a four steps process:

1. The client sends a HTTPS/REST "GET **Mission Plan**" request with selection parameters;
2. In response, the server returns the URI and version of thefile containing the requested plan;
3. The client checks the version against the version of its latest download. If needed, the client downloads the file;
4. In response, the server returns the requested file.



[mission\_plan.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/327975075/mission_plan.pptx?api=v2&modificationDate=1687270290000&version=1)

Typical client/server:

|  |  |
| --- | --- |
| Clients | Server |
| FOS | MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **Mission Plan** request | Asks for a mission plan for a satellite and over a period | satellite,    period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **Mission Plan** response | 200 | in case of success, the URI and version of the file containing the mission plan | {    uri,    version  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **File** request | Requests the file for the given URI | URI |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **File** response | 200 | in case of success, returns the mission plan as a list of Time Tagged TC in an abstract language | {     metadata,    "Time-tagged TC":    [      "event": { eventClass, eventName, time, parameters }    ]  } |

*Note on status codes:*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data types

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| **PARAMETERS** |  |  |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| uri | String | File identifier |
| version | String | Version (or tag) of the file |
| **MISSION PLAN** |  |  |
| metadata | JSon object | metadata about the plan (satellite, horizon, production date...) |
| eventClass | String | a category for the event |
| eventName | String | the name of the event |
| time | Date | the planned execution time |
| parameters | JSon array | the parameters of the event |

### Example

Time-tagged TC example:

    {

      "eventClass": "thrustEvent",

      "eventName": "startThrust",

      "time": "2023-07-02T00:00:00.000",

      "parameters": [

        {

          "name": "level",

          "value": "low"

        }

      ]

    }

## Mission Reprogramming Request

### Introduction

The "**Mission Reprogramming Request**" Interface allows clients to retrieve a list of mission reprogramming requests since a date (till today) .

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requierements :

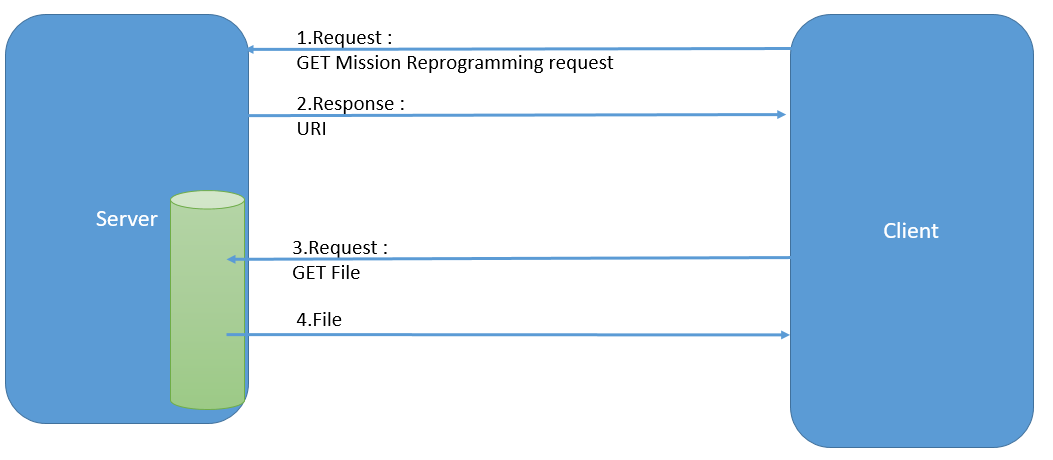
* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

The supported format for designing the API is JSON.

### Dynamics of exchanges

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **Mission Reprogramming Request**" request with a date as parameter.
2. In response, the server returns a URI (link to a file containing a list of Mission Reprogramming Requests).
3. Then, the client select this URI to download this file.
4. In response, the server return a Mission Reprogramming Request file (file containing GUID User Requests with theirs conditionalProductRequests)



[Mission Reprogramming Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641131/Mission%20Reprogramming%20Request.pptx?api=v2&modificationDate=1679926190000&version=2)

Client/server example :

|  |  |
| --- | --- |
| Cients | Servers |
| FS | KBDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **Mission Reprogramming Request** request | Ask for a list of mission reprogramming requests since a date (till today) | period (by default the end date is the current date) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Mission Reprogramming Request** response | 200 | Successfully returned an URI (link to a file containing a list of reprogramming requests) | {  URI (link to a file containing a list of reprogramming requests  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a mission reprogramming request file. | [   { userRequest guid     conditionalProductRequest {        programmingRequest,        productionRequest,        deliveryRequest     }   },  …]  *Note :  The conditionalProductRequest is given into the response file but is strictly the same as the one configure by the End User in its User Request. It is given into the response file to be compatible of future evolution of the KBDS (to be able to update it if needed).* |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| userRequest guid | Integer | User request guid |
| conditionalProductRequest | ConditionalProductRequest | Conditional Product Request  (see [User Request Activation Request](#scroll-bookmark-54)) |

## Observable

### Introduction

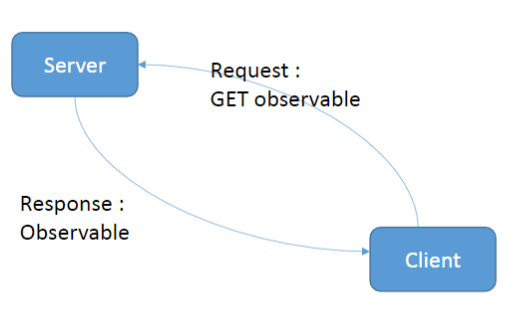
Observable are data or raw KPI used to understand a system’s internal state. They are used to displaying data on HMI or to generate report.

Due to the need of some observable for HMI display, polling frequency can be high but will not exceed 1 query every 5 seconds (refresh rate considered acceptable for OMS HMI)

### Dynamics of exchanges

Periodically, the OMS sends an HTTPS/REST "**GET Observable**" request to the dominoes. In response, the requested domino computes and returns the requested monitoring information.

This period is changing depending on the data usage and so can increase or decrease. For example: if the HMI, that displayed these observable, is switched off, the recovery of the observable feeding them is no longer necessary and can be paused.



[Observable.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640679/Observable.pptx?api=v2&modificationDate=1680252272000&version=1)

These Web Service are based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Flow details

#### Request

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| **GET Observable** request | Ask for current observable value | * **Ids**: list of requested observable |

#### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | **Code Status** | **Description** | **JSON response** |
| **Observable** response | 200 | List of observable values | A list of observable. Each observable is characterized by the following fields:   * + **ID** observable identifier.   + **Date:** date of the observable value   + **Value** observable value |
| **Observable** response | 206 | Same than HTTP 200 response used when, at least, a requested ID is unknown by the domino. | A list of observable. Each observable is characterized by the following fields:   * + **ID** observable identifier.   + **Date:** date of the observable value   + **Value** observable value (value is null for unknown observable) |

#### Data type

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Type | Values | Description |
| Date | MM/DD/YY HH:MM:SS |  | Date |
| ID | String |  | format: **TBD** |
| Value | Value can be one of the 6 next data types | String | An alphanumeric sequence of letters and/or numbers |
| Number | A whole number (can be positive or negative) |
| Boolean | True or False |
| Object | Key-Value pair |
| Array | A list of Value |
| Null | Undefined or unknown observable |

## On-board report

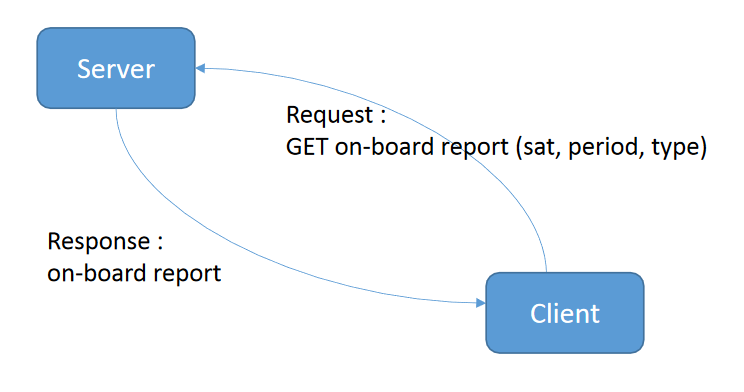
### Introduction

The "**On-board Report**" Interface allows clients to get information about the on-board state and also about the upload state of a mission plan or a maneuver plan.

What kind of information is returned depends on the requested report type.

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **On-board Report**" request with selection parameters. In response, in case of success, the server returns a report.



[onboard\_report.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/327975081/onboard_report.pptx?api=v2&modificationDate=1687261903000&version=1)

Typical client/server:

|  |  |
| --- | --- |
| Clients | Server |
| FDS, MPS, IQS | FOS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **On-board Report** request | Asks for a report of a given kind,  for a satellite and over a period | Query Parameters    satellite,    period,    reportType (ON\_BOARD\_STATE, PLAN\_UPLOAD, PAYLOAD\_CALIBRATION\_UPLOAD) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **On-board Report** response | 200 | in case of success, the requested report | {    satellite,    period,  *<details>*  }  where *<details>* depends on *reportType*:  if reportType is ON\_BOARD\_STATE :  *<details>*=modes : [    {    satelliteMode (ie. NOMINAL, ANOMALY, UNKNOWN),    period    },    …]  if reportType is PLAN\_UPLOAD or PAYLOAD\_CALIBRATION\_UPLOAD :  *<details>*=uploads : [    {    uploadDate,    planId,    status (OK, KO)    },    …] |

*Note on status codes:*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data types

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| reportType | String | kind of report. One of: ON\_BOARD\_STATE, PLAN\_UPLOAD |
| satelliteMode | String | kind of on-board state. One of: NOMINAL, ANOMALY, UNKNOWN |
| uploadDate | Date | latest date when an attempt to load the plan was made |
| planId | String | id of the plan for which a load attempt was made |
| status | String | status of the upload attempt. One of: OK, KO |

### Example

Request:

{

    "satellite": "Sputnik",

    "period": {

      "startDate": "2021-07-02T00:00:00.000",

      "endDate": "2021-07-02T24:00:00.000"

    },

    "reportType": "ON\_BOARD\_STATE"

}

Response for ON\_BOARD\_STATE:

{

    "satellite": "Sputnik",

    "horizon": {

      "startDate": "2021-07-02T00:18:59.127",

      "endDate": "2021-07-02T23:34:16.512"

    },

    "modes": [

      {

        "satelliteMode": "NOMINAL",

        "period": {

          "startDate": "2021-07-02T00:18:59.127",

          "endDate": "2021-07-02T13:00:00.000"

        }

      },

    …

    ]

}

Response for PLAN\_UPLOAD:

{

    "satellite": "Sputnik",

    "horizon": {

      "startDate": "2021-07-02T00:18:59.127",

      "endDate": "2021-07-02T23:34:16.512"

    },

    "uploads": [

      {

        "uploadDate": "2021-07-02T13:00:00.000",

        "planId": "a1980658-0f63-11ee-be56-0242ac120002",

        "status": "OK"

      },

    …

    ]

}

## Orbit Events

### Introduction

The "**Orbit Events**" Interface allows clients to get information about specific moments a satellite meets on its orbit.

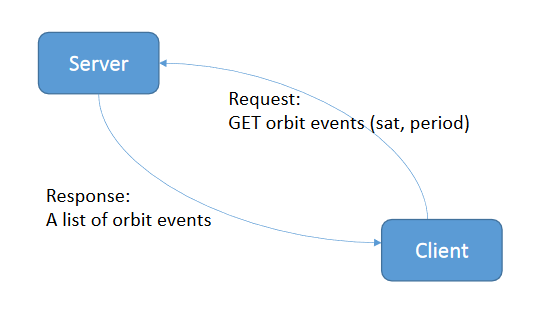
This Web Service is based on OPENAPI 3.0 standard interface. The interface payload format is JSON.

Typical events are:

* Passage to apogee and perigee;
* Passage through the ascending and descending nodes;
* Passage through the sub-solar and anti-solar points;
* Passage through the 90° and 270° PSO points ("poles" of the orbit);
* Passage near the poles (at configurable latitude);
* SAA inputs/outputs;
* Earth or Moon eclipses:
  + Transition shadow to penumbra;
  + Transition penumbra to day;
  + Transition day to penumbra;
  + Transition penumbra to shadow.

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **Orbit Events**" request with selection parameters. In response, in case of success, the server returns a list of orbit events.



[orbit\_events.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/327975073/orbit_events.pptx?api=v2&modificationDate=1686665384000&version=2)

Typical client/server:

|  |  |
| --- | --- |
| Clients | Server |
| FOS | FDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **Orbit Events** request | Asks for a list of events for a satellite and over a period | Query Parameters:    satellite,    period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **Orbit Events** response | 200 | in case of success, the orbit events data | {    horizon: Period,    events: [      {date, eventClass: {category, type, label}},      …    ]  } |

*Note on status codes:*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data types

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| period | Period | Start date, end date (availability time spans) |
| satellite | String | Satellite name |
| date | Date | moment when the event should occur |
| category | String | a classification of the event (ORBITAL, STATION, ...) |
| type | String | a distinction on the nature of the event (ECLIPSE, APSIS, NODE, FLYOVER, ...) |
| label | String | name of the event |

### Example

Request:

{

    "satellite": "Sputnik"

    "period": {

      "startDate": "2021-07-02T00:00:00.000",

      "endDate": "2021-07-02T24:00:00.000"

    }

}

Response:

{

    "horizon": {

      "startDate": "2021-07-02T00:18:59.127",

      "endDate": "2021-07-02T23:34:16.512"

    },

    "events": [

      {

        "date": "2021-07-02T00:18:59.127",

        "eventClass": {

          "category": "ORBITAL",

          "type": "APSIS",

          "label": "PERIGEE"

        }

      },

…

}

## Payload Calibration

### Introduction

The "**Payload Calibration**" Interface allows clients to retrieve a list payload calibration files for a given constellation, satellite and period.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requierements :

* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

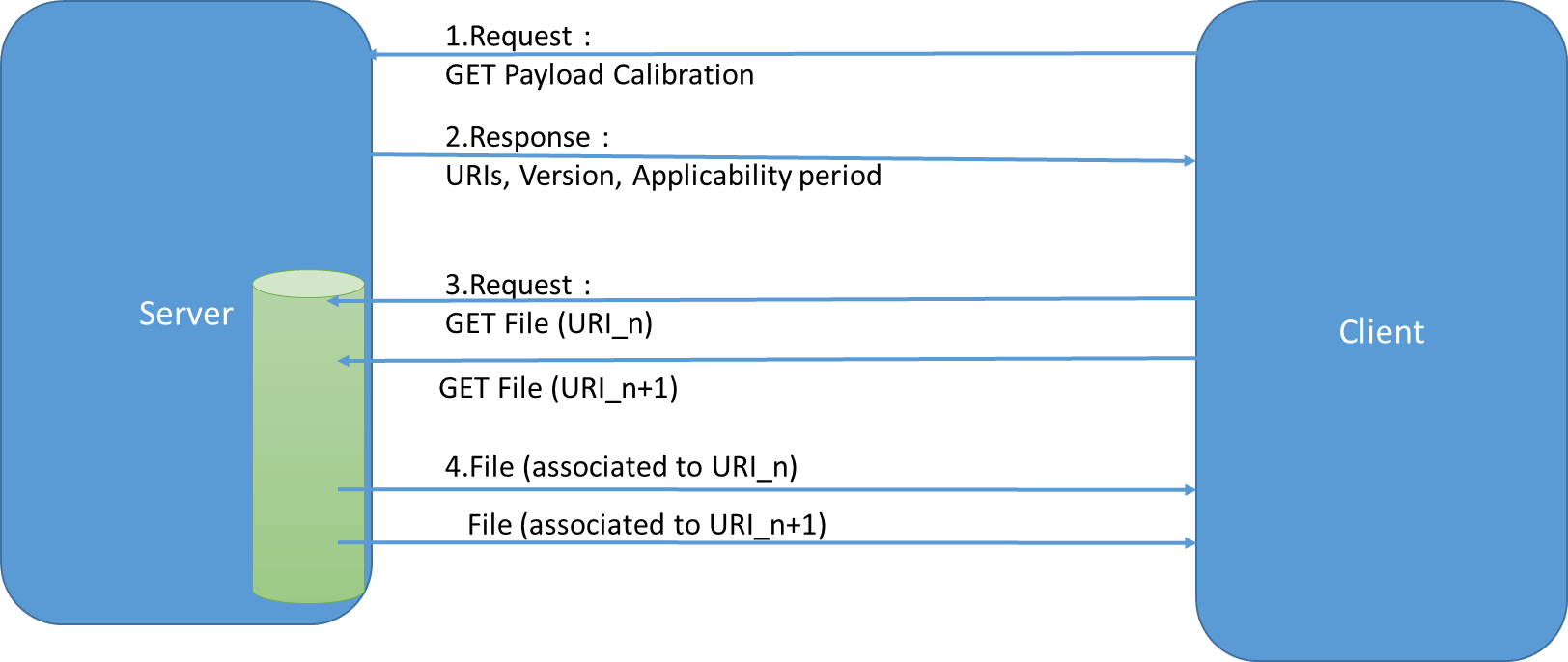
These files are mission specific, so no generic format is defined but JSON is recommanded.

### Dynamics of exchanges

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **Payload Calibration**" request with constellation, satellite and period as parameters.
2. In response, the server returns a list of URIs of files with their version and applicability period.
3. Then, among all the URIs available (those transmitted in step 2), the client chooses the files (those associated with the chosen URIs) that he really wishes to receive. The client sends a HTTPS/REST "GET File"with URI as parameters to retrieve one file.
4. In response, the server returns a file.

[Payload\_calibration.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/329680003/Payload_calibration.pptx?api=v2&modificationDate=1687349488000&version=1)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| FOS | IQS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Payload Calibration**request | Ask for a list of URIs files for a given constellation, satellite and applicability date. | Query Parameters:     constellation     satellite     applicability date |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Payload Calibration** response | 200 | Successfully returned a list of URIs of corresponding files with version and applicability period. | URIs:[  {     URI (link to download the files)     version     period  }, …  ] |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI (one by file) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a file | Mission specific format. JSON recommanded |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| comment | String | Comment |
| version | String | Version (or tag) of the file |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| uri | String | File identifier |

## Polarisation Plan

### Introduction

The "**Polarisation Plan**" Interface allows clients to retrieve a list polarisation plan files for a given constellation, satellite and period.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requierements :

* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

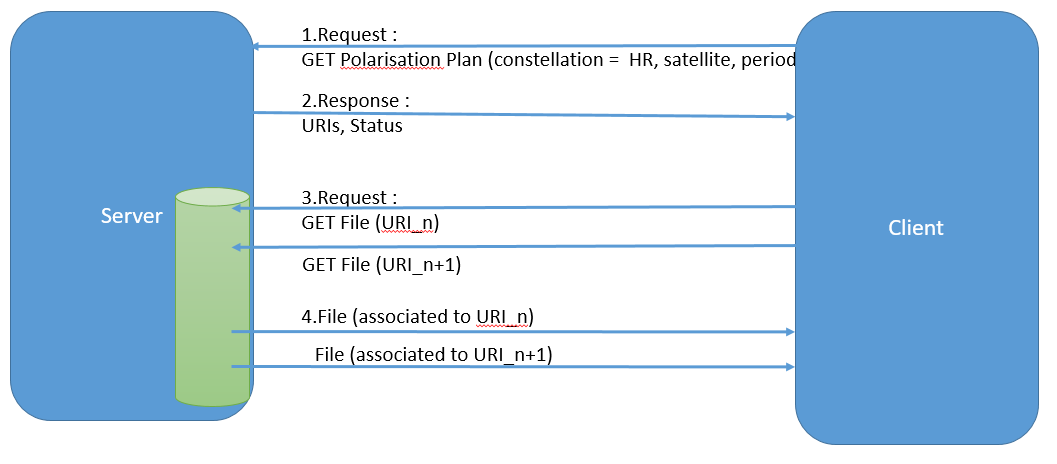
The supported format for designing the API is JSON.

### Dynamics of exchanges

[Polarisation\_Plan.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397196/Polarization_Plan.pptx?api=v2&modificationDate=1678888831000&version=2)

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **Polarisation Plan**" request with constellation, satellite and period as parameters.
2. In response, the server returns a list of URIs of polarisation plan files with their completion status (complete, partial, or with error)
3. Then, among all the URIs available (those transmitted in step 2), the client chooses the files (those associated with the chosen URIs) that he really wishes to receive. The client sends a HTTPS/REST "GET File"with URI as parameters to retrieve one file.
4. In response, the server returns a polarisation plan file.



[Polarisation\_Plan.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397196/Polarisation_Plan.pptx?api=v2&modificationDate=1680094723000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Cients | Servers |
| FOS | SCRMS |
| SCRMS | MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Polarisation Plan** request | Ask for a list of URIs polarisation plan files for a given constellation, satellite and period. | Query Parameters     constellation = HR     satellite     period  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Polarisation Plan** response | 200 | Successfully returned a list of URIs of polarisation plan files with a completion status (COMPLETE, PARTIAL, ERROR) and comment. | List of URIs ( for example URIs of AOCS or applicable polarisation plan files)  with a completion status and comment :  {  URIs (List of URIs : one URI by file),  completion\_status (COMPLETE, PARTIAL, ERROR),  comment  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI (one by file) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a polarisation plan file. | Polarisation plan file :  [    constellation     satellite     {     period     polarisation    }     {     period     polarisation    }  Etc.  ] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| comment | String | Explanation in case of partial error status. |
| completion status | Enum of strings | Complete, partial, error |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| polarisation | Enum of strings | Satellite polarisation (right /  left) |
| satellite | String | Satellite name |
| uri | String | File identifier |

## Predicted Orbits

### Introduction

The "**Predicted Orbits**" Interface allows clients to retrieve a list predicted orbit files for a given constellation, satellite and period.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requierements :

* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

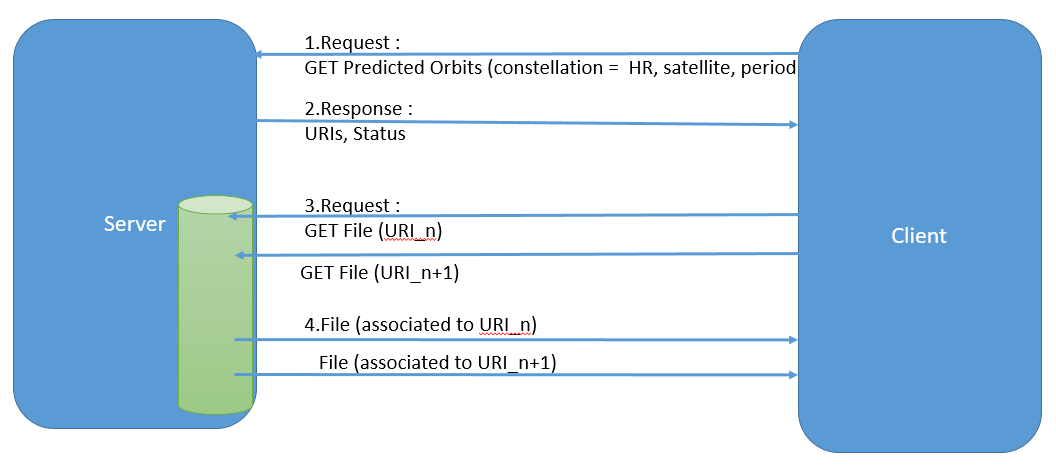
The supported format for designing the API is JSON.

### Dynamics of exchanges

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **Predicted Orbits**" request with constellation, satellite and period as parameters.
2. In response, the server returns a list of URIs of polarization plan files with their completion status (complete, partial, or with error)
3. Then, among all the URIs available (those transmitted in step 2), the client chooses the files (those associated with the chosen URIs) that he really wishes to receive. The client sends a HTTPS/REST "GET File"with URI as parameters to retrieve one file.
4. In response, the server returns a predicted orbit file.

[Predicted\_Orbits.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307397400/Predicted_Orbits.pptx?api=v2&modificationDate=1678889263000&version=2)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| SCRMS | FDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Predicted Orbits** request | Ask for a list of URIs predicted orbit files for a given constellation, satellite and period. | Query Parameters:     constellation = HR     satellite     period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Predicted Orbits** response | 200 | Successfully returned a list of URIs of predicted orbit files with a completion status (COMPLETE, PARTIAL, ERROR) and comment. | List of URIs ( URIs of predicted orbit files)  with a completion status and comment :  {  URIs (List of URIs : one URI by file),  completion\_status (COMPLETE, PARTIAL, ERROR),  comment  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI (one by file) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a predicted orbit file | Predicted orbit file :  [     constellation     satellite     {     period     precise Orbit Information    }     {     period     precise Orbit Information    }  Etc.  ] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| comment | String | Comment |
| completion status | Enum of strings | Complete, partial, error |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| precise orbit information | EphemerisList | List of ephemeris |
| satellite | String | Satellite name |
| uri | String | File identifier |

## Product Availability Notification to be consolidated (pick-up-point-light)

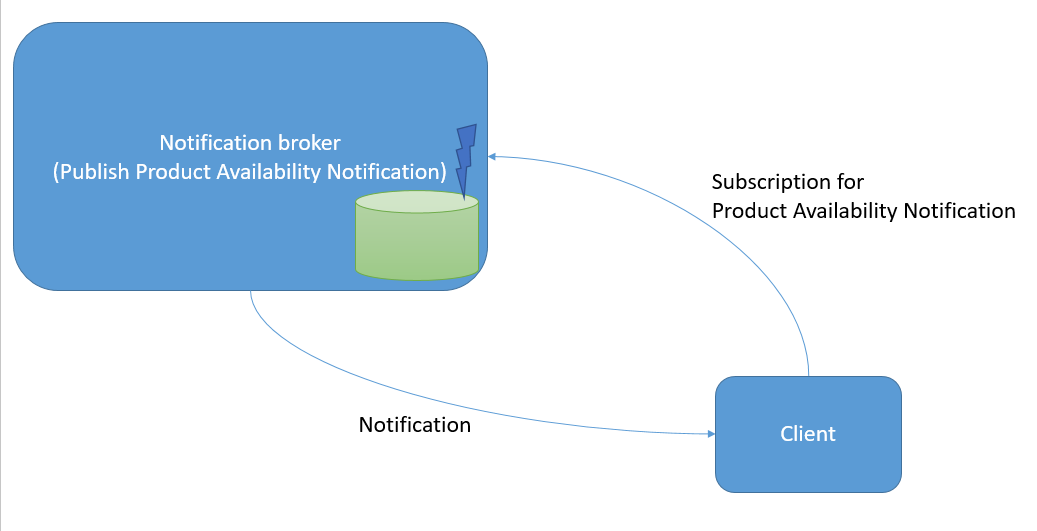
### Introduction

The "**Product availability Notification**" Interface allows clients to subscribe to a broker notification broker in order to receive a notification when a product is available.

This subscription is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "TBD **Product availability Notification**" request with a product type and event type as parameters (TBD = the communication protocol with the notification broker is not defined. It will be set when parsing [UAC]).

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641095/ProgR%20Consultation%20Request.pptx?api=v2&modificationDate=1679919728000&version=1)

[Product Availability Notification.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641095/Product%20Availability%20Notification.pptx?api=v2&modificationDate=1679919856000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server (notification broker) |
| FS | ACS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| TBD **Product availability Notification**request | Subscribe to a broker notification in order to receive a notification when a product is available. | {  product type  event type (creation , deletion, update)  } |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| Notificationresponse | 200 | When a product is available | {     product ID     product path     product metadata  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| event type | Enum of strings | Creation, deletion, update |
| product ID | Integer | Product Id |
| product metadata | ProductMetaData | Product metadata |
| product path | String | Product path |
| product type | Enum of strings | Product type (primary, ortho, etc.) |

## Product Integrity Verification Request

### Introduction

The "**Product** **Integrity Verification Request**" Interface allows clients (i.e dominoes meant to ingest products) to verify the integrity of the product received before launching their processing actions by computing locally the hashcode of the product and submitting it to the centralized DITS domino for verification.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

|  |  |
| --- | --- |
| Clients | Server |
| ACS | DITS |
| PPS | DITS |
| APS | DITS |
| EPS | DITS |
| DDS | DITS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| POST **Product** **Integrity Verification Request** | Sends a locally computed hascode for a product for verification that the product is known with the same integrity reference value. |  |

|  |  |  |
| --- | --- | --- |
| **Interface** | **Status code** | **Description** |
| POST **Product** **Integrity Verification Request** response | 200 | The triplet <productId/productHashType/productHash> is properly recognized |
|  | 400 | The provided hash code does not correspond to the registered hashcode (error code: \*\*1003\*\*) |
|  | 404 | The product (error code: \*\*1001\*\*) or the product Hash Type (error code: \*\*1002\*\*) was not recognized |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |

## Production Context

### Introduction

The "**Production Context**" Interface allows clients to retrieve a context for a given production.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requirements :

* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

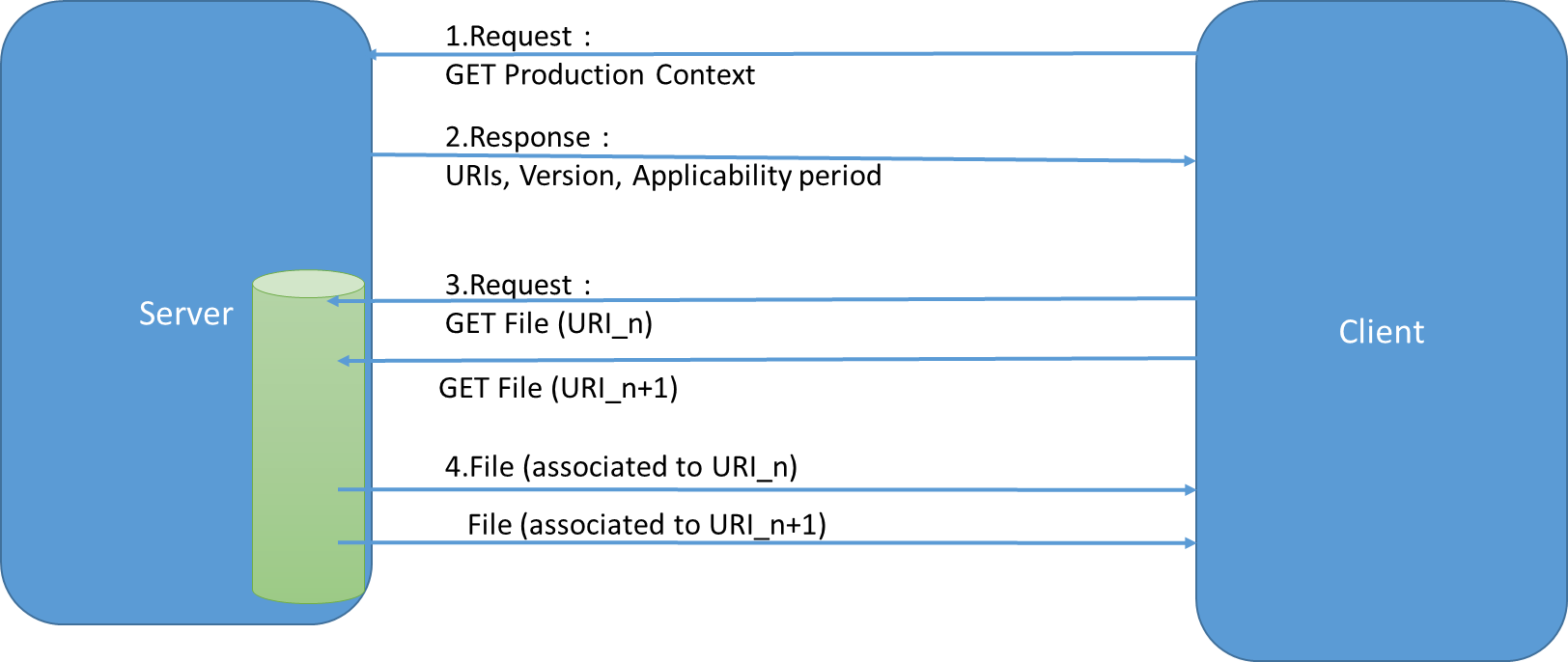
These files are mission specific, so no generic format is defined but compressed archive is recommanded.

### Dynamics of exchanges

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **Production Context**" request with constellation, satellite and period as parameters.
2. In response, the server returns a list of URIs of files.
3. Then, among all the URIs available (those transmitted in step 2), the client chooses the files (those associated with the chosen URIs) that he really wishes to receive. The client sends a HTTPS/REST "GET File"with URI as parameters to retrieve one file.
4. In response, the server returns a file.

[Production\_context.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/329680033/Production_context.pptx?api=v2&modificationDate=1687349669000&version=1)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| IQS | PPS/APS |
| MAPQS | EPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Production Context** request | Ask for a list of URIs files for a given productID. | Query Parameters:       Constellation     Satellite     ApplicabilityDate     ContextLevel |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Production Context** response | 200 | Successfully returned a list of URIs of corresponding context(s). | URIs:[  {     URI (link to download the files)  }, …  ] |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI (one by file) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a file | Compressed archive. |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| Constellation | String |  |
| Satellite | String |  |
| ApplicabilityDate | String |  |
| productID | String |  |
| context level | String | How precise the context must be |

## Products to sign

## Product Traceability Request

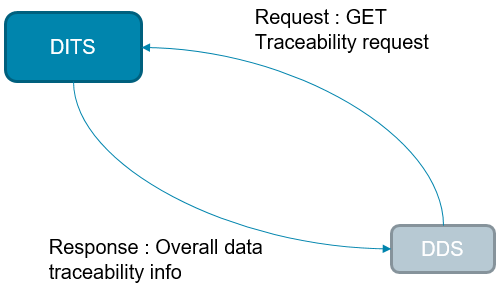
### Introduction

The "**Product Traceability Request**" Interface allows clients to retrieve the full traceability of a product at delivery.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for the API payload is JSON.

### Dynamics of exchanges

The DDS sends a HTTPS/REST "GET **Product Traceability Request**" request with the product ID to be downloaded as parameter to the DITS.



Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| DDS | DITS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| **GET Product Traceability Request** request | Ask for the full traceability of a product. | traceability request {       productID      } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Product Traceability Request**response | 200 | Successfully returned the full traceability | HTTPS status |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| traceability request | TraceabilityRequest |  |

## ProgR Activation Request

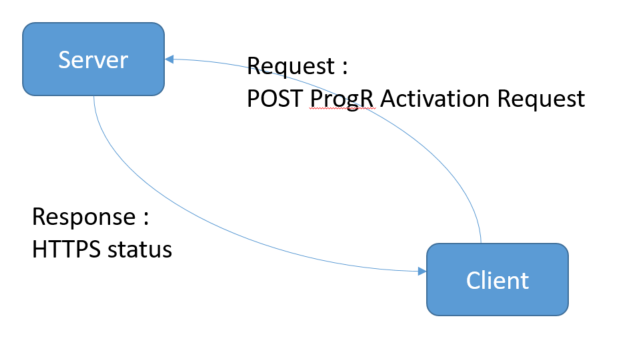
### Introduction

The "**ProgR Activation Request**" Interface allows clients to activate a "Programming Request".

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **ProgR Activation Request**" request with a Programming Request as parameter (see [User Request Activation Request](https://wiki-external.thalesaleniaspace.fr/display/DEOP/User+Request+Activation+Request?src=contextnavpagetreemode)).

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640821/ProgR%20Analysis%20Request.pptx?api=v2&modificationDate=1679909350000&version=1)

[ProgR Activation Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640821/ProgR%20Activation%20Request.pptx?api=v2&modificationDate=1679910273000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | INTS/MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **ProgR Activation Request**request | Ask to activate a programming request | programming request {       ... (see [User Request Activation Request](#scroll-bookmark-54))      } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **ProgR Activation Request**response | 200 | Successfully returned a status | HTTPS status |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| programming request | ProgrammingRequest | see [User Request Activation Request](#scroll-bookmark-54) |

## ProgR Analysis Request

### Introduction

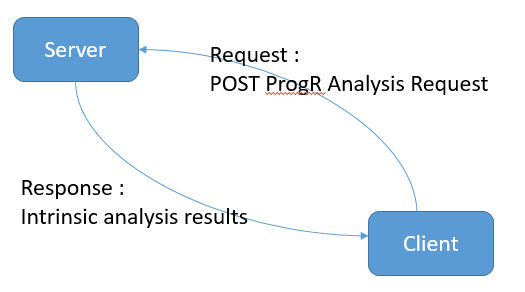
The "**ProgR Analysis Request**" Interface allows clients to consult the intrinsic analysis results of a Programming Request (ProgR).

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **ProgR Analysis Request**" request with a Programming Request as parameter (see [User Request Activation Request](https://wiki-external.thalesaleniaspace.fr/display/DEOP/User+Request+Activation+Request?src=contextnavpagetreemode)).

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907161/ProgR%20Analysis%20Request.pptx?api=v2&modificationDate=1680074345000&version=2)

[ProgR Analysis Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907161/ProgR%20Analysis%20Request.pptx?api=v2&modificationDate=1680074345000&version=2)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **ProgR Analysis Request** request | Ask to consult to consult the intrinsic analysis results of a Programming Request | programming request {       ... (see [User Request Activation Request](#scroll-bookmark-54))      } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **ProgR Analysis Request**response | 200 | In case of success return an intrinsic analysis results of a Programming Request | [{       constellation ,       [acquisition request {             meshes [ {guid,aoi}, … ]              psoDTOs [{guid, satellite, orbital\_direction, orbit\_number, period, }, … ]              datedDTOs [{guid, satellite, orbital\_direction, period},…]                } ,  …]  },   …] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Constellation |
| meshes | Meshes | List of areas of Interest |
| psoDTOs | PsoDTOs | Position sur orbite (orbit position) Data Take Opportunities |
| datedDTOs | DatedDTOs | Dated Data Take Opportunities ( Dated shooting opportunities) |

## ProgR Cancel Request

### Introduction

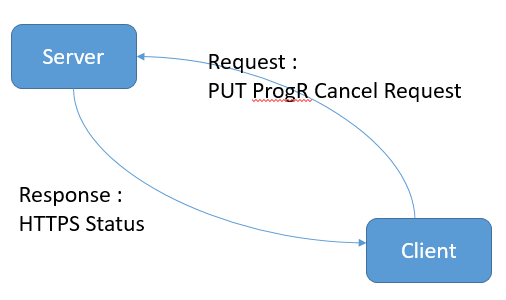
The "**ProgR Cancel Request**" Interface allows clients to cancel a Programming Request.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "PUT **ProgR Cancel Request**" request with a GUID Programming Request and CANCELLED status as parameters.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641027/ProgR%20Consultation%20Request.pptx?api=v2&modificationDate=1679917389000&version=1)

[ProgR Cancel Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641027/ProgR%20Cancel%20Request.pptx?api=v2&modificationDate=1679917900000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | INTS/MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| PUT **ProgR Cancel Request**request | Ask to cancel a Programming Request | {     guid (of a programming request)     status = CANCELLED  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| PUT **ProgR Cancel Request** response | 204 | return HTTPS status | HTTPS status |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| guid | Integer | GUID Programming Request |
| status | Integer | HTTPS status code |

## ProgR Consultation Request

### Introduction

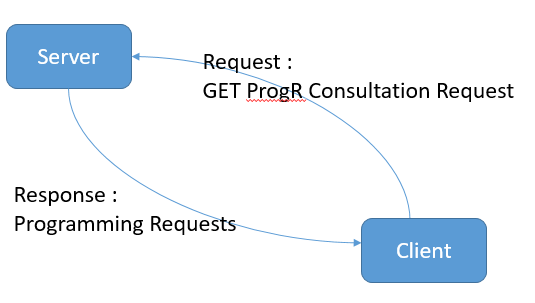
The "**ProgR Consultation Request**" Interface allows clients to consult a list of Programming Requests according to search criteria.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **ProgR Consultation Request**" request with search criteria as parameters.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640970/ProgR%20Consultation%20Request.pptx?api=v2&modificationDate=1680015592000&version=2)

[ProgR Consultation Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640970/ProgR%20Consultation%20Request.pptx?api=v2&modificationDate=1680015592000&version=2)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **ProgR Consultation Request**request | Ask to consult a list of Programming Requests according to search criteria | Query Parameters:   browsing criteria (attribute | logical operator | values) expected attributes (in response) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **ProgR Consultation Request**response | 200 | In case of success return a list of "Programming Requests" | [  programming request {  GUID,  ....  },  ...  ] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| browsing criteria | BrowsingCriteria | Browsing critera (attribute , logical operator, values, etc.) |
| programming request | ProgrammingRequest | Programming Request (see [User Request Activation Request](#scroll-bookmark-54) Interface) |

## ProgR Update Request

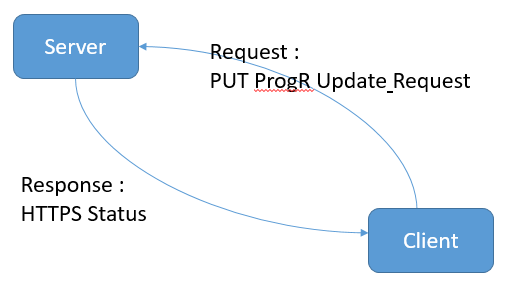
### Introduction

The "**ProgR Update Request**" Interface allows clients to update a Programming Request.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "PUT **ProgR Update Request**" request with Programming Request as parameter.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641055/ProgR%20Update%20Request.pptx?api=v2&modificationDate=1680016413000&version=2)

[ProgR Update Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310641055/ProgR%20Update%20Request.pptx?api=v2&modificationDate=1680016413000&version=2)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| FS | INTS/MPS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| PUT **ProgR Update Request** request | Ask to update a Programming Request | programming request  {     guid (of a programming request)     ... (updates)  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| PUT **ProgR Update Request**response | 204 | return HTTPS status | HTTPS status |

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| programming request | ProgrammingRequest | Programming Request ( see [User Request Activation Request](#scroll-bookmark-54)) |
| status | Integer | HTTPS status code |

## Routine Pivot Point Slots

### Introduction

The "**Routine Pivot Point Slots**" Interface allows clients to straightforwardly retrieve a list of RPP slots (slots used for mission plan transitions) and that for a given constellation, satellite and period.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON

### Dynamics of exchanges

The client sends a HTTPS/REST "GET "**Routine Pivot Point Slots**" request with constellation, satellite and period as parameters . In response, in case of success, the server returns a list of RPP slots over the period given as input to the request.

Client/servers example :

|  |  |
| --- | --- |
| Clients | Server |
| MPS | SCRMS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Unavailability Slots** request | Ask for a list of unavailability slots for a given constellation, satellite and period. | Query Parameters:     constellation = HR     satellite     period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Unavailability Slots** response | 200 | Successfully returned a list of unavailability slots | List of unavailability slots :  {     constellation     satellite     RPP period  }  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| RPP period | Period | Start date, end date (time slots over the period given as input to the request) |

## Signed products

## Site Monitoring Allocation Projection Request

### Introduction

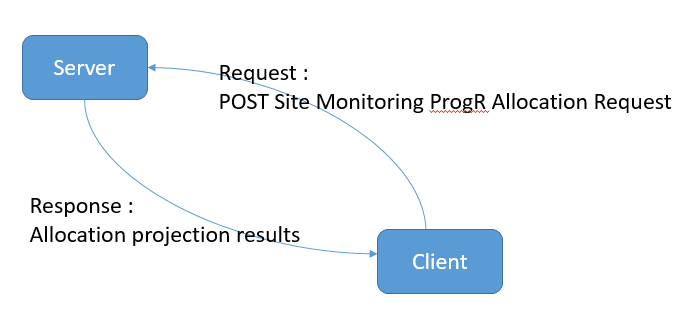
The "**Site Monitoring Allocation Projection Request**" Interface allows a client to study the feasibility of site monotoring allocation projection request. This request returns a allocation projection results.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **Site Monitoring Allocation Projection Request**" request with a user request as parameter. This user request contains a system programming configuration.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640731/Coverage%20ProgR%20Manual%20Reassesment.pptx?api=v2&modificationDate=1679904100000&version=1)

[Site Monitoring Allocation Projection Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/310640731/Site%20Monitoring%20Allocation%20Projection%20Request.pptx?api=v2&modificationDate=1679909219000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **Site Monitoring Allocation Projection Request**request | On operator demand, ask to study the feasibility of site monotoring allocation request. | user request  {     ...  (see [User Request Activation Request](#scroll-bookmark-54))     periodicityParameters        system\_prog\_configuration [           { constellation             scoringParameters             splitParameters             acquisitionParameters             angularConstraints           }, …  ]  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **Site Monitoring Allocation Projection Request**response | 200 | In case of success return a allocation projection results | {       is\_feasible,       dispatch\_result  [         { selected\_sat, access\_start, access\_end },…    ]  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| access end | Date | End access date |
| access start | Date | Start access date |
| acquisition parameters | AcquisitionParameters | Acquisition parameters |
| angular contraints | AngularConstraints | Angular contraints |
| constellation | String | Satellite constellation name |
| is feasible | Boolean | True/False |
| periodic parameters | PeriodicParameters | Periodic Parameters |
| scoringParameters | Scoring Parameters | Scoring Parameters |
| selected sat | String | Selected satellite |
| splitParameters | SplitParameters | Split Parameters |

## System and Satellite Configuration

### Introduction

The "**System and Satellite Configuration**" Interface allows clients to retrieve the system and satellite configuration for a given satellite.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requierements :

* Asynchronous Exchange
* Centralized data
* One pickup-point for each data producer
* Data described in file
* Etc.

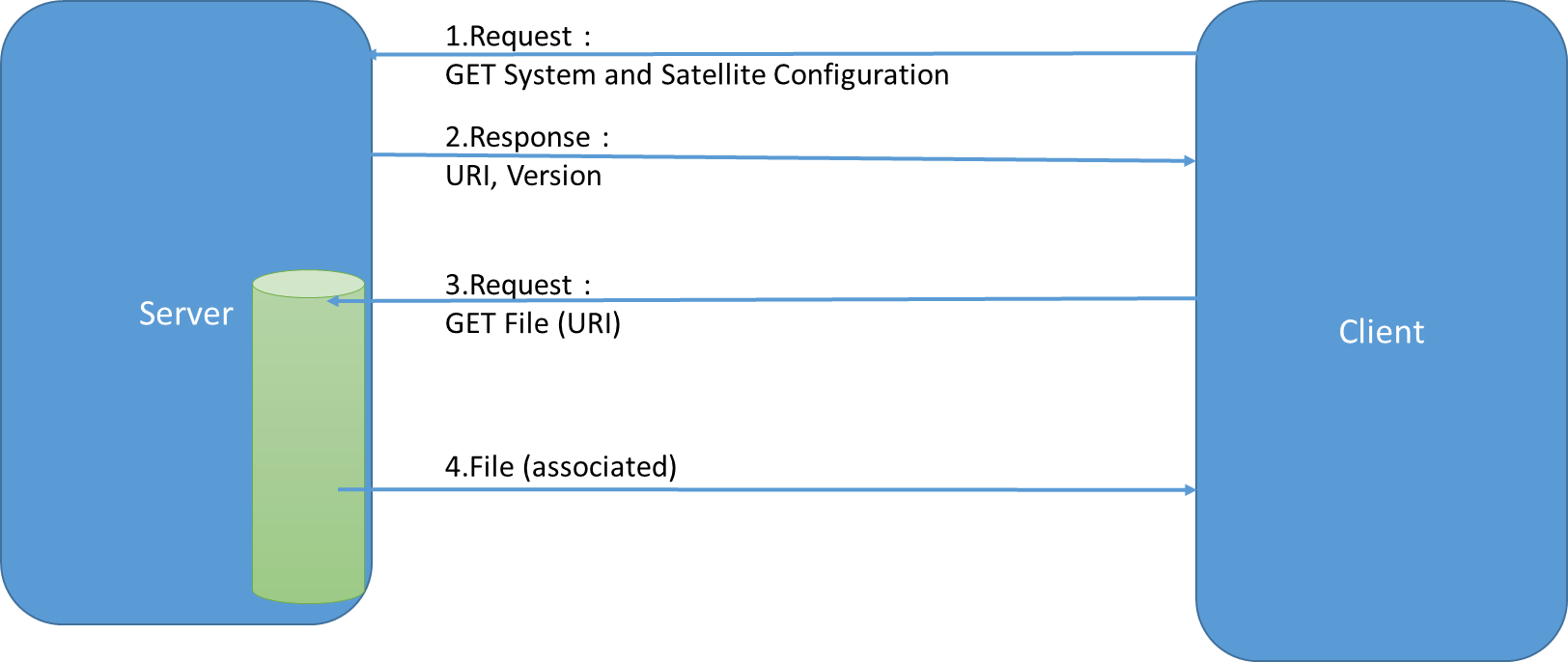
The supported format for designing the API is JSON.

### Dynamics of exchanges

The exchange takes place in four steps :

1. The client sends a HTTPS/REST "GET **System and Satellite Configuration**" request with constellation, satellite and applicability date, and consumer as parameters.
2. In response, the server returns the URI ofSystem and Satellite Configuration file  with the associated version
3. The client will check if it already have the given version, if so no download is needed, if not the file is downloaded by the client.
4. In response, the server returns a  system and satellite configuration file.

[system\_sat\_conf.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/326369862/system_sat_conf.pptx?api=v2&modificationDate=1686312996000&version=1)



Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| MPS, FOS | SCMS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **System and Satellite Configuration**request | to retrieve the system and satellite configuration for a given satellite. | Query Parameters:   * Constellation * Satellite * Consumer = MPS * Applicability date |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **System and Satellite Configuration** response | 200 | Successfully returned a URI of the corresponding file. | {     URI (link to download the file)     version (of the SSC)  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **File** request | Ask for a file corresponding to the URI sent in parameter | URI (one by file) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a predicted orbit file | {  constellation  Satellite  [params]  }  Note : The parameters are specific to a mission/constellation hence not described here |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| uri | String | File identifier |
| version | String | Version (or tag) of the file |

## TM Values

### Introduction

The "**TM Values**" Interface allows clients to get the values of TM parameters over a given period.

This Pickup Point is based on OPENAPI 3.0 standard interface. It responds to Pickup Point requirements:

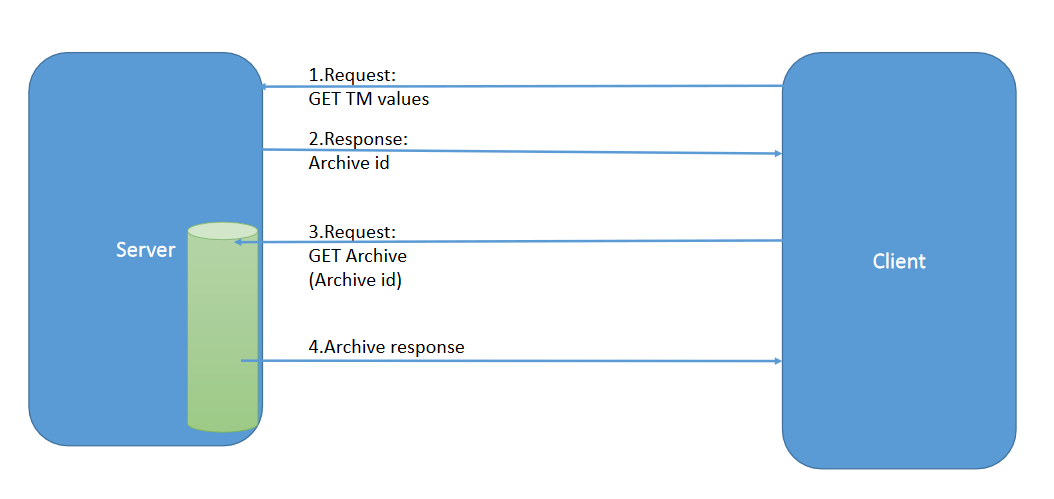
* Asynchronous Exchange;
* Centralized data;
* One pickup-point for each data producer;
* Data described in a file.

The interface format is JSON.

### Dynamics of exchanges

The exchange is a several steps process:

1. The client sends a HTTPS/REST "GET **TM Values**" request with selection parameters;
2. In response, the server returns the archiveId of thearchive containing the requested TM values;
3. The client downloads the archive;
4. In response, the server returns the requested archive;
5. As long as the status is not 'ready', the client waits a while and requests again.



[tm\_values.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/327975083/tm_values.pptx?api=v2&modificationDate=1687266116000&version=1)

Typical client/server:

|  |  |
| --- | --- |
| Clients | Server |
| FDS, satellite integrator | FOS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **TM Values** request | Asks for TM values for a satellite and with selection parameters | {    satellite,    chain,    sampling,    filters,    parameters,    period } |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **TM Values** response | 200 | in case of success, the archiveId of the file containing the TM values | {    Archive id  } |

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Parameters** |
| GET **Archive** request | Requests the file for the given archiveId | Archive id |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **Response** |
| GET **Archive** response | 200 | in case of success, returns the list of requested TM values along with the status of the archive | {     archiveStatus (in progress, ready),    "data":    [{      "groundTime",      "parameters" : [ parameter ]    }]  } |

*Note on status codes:*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data types

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| **PARAMETERS** |  |  |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| chain | String | kind or category of TM |
| sampling | String | either "all" "onchange", or an integer to define the frequency of the sampling on a per item basis |
| filters | Array of strings | type of information for each value. One or several of: "extracted" "raw" "physical" "tostring" "state" "validity" "obt" |
| parameters | Array of strings | the names of the parameters to extract |
| **TM values** |  |  |
| archiveStatus | String | status of the response readyness. One of: in-progress, ready. |
| groundTime | Date | time the parameter was valued, in ground time reference |
| parameter | JSon object | the values for a parameter at a given moment.  The contents depend on the value of the filters specified in the request |
| **Parameter fields** |  |  |
| id | String | parameter name |
| xvalue | Array of strings | extracted value |
| rvalue | String | raw value |
| pvalue | String | physical value |
| dvalue | String | display value |
| status | Array of integers | state of the value |
| obtvalue | Date | on-board time |

### Example

Request:

{  
  "satId": "Sputnik",  
  "chain": "hktmp",  
  "sampling": 4,  
  "filters": [  
    "physical",  
    "obt"  
  ],  
  "parameters": [  
    "SSYS\_BP\_PUSSERVICE",  
    "SSYS\_BP\_PUSSUBTYPE"  
  ],  
    "period": {  
      "startDate": "2021-07-02T00:00:00.000",  
      "endDate": "2021-07-02T24:00:00.000"  
    }  
}

TM values archive:

{

  "archiveStatus": "ready",

  "data": [

    {

      "groundTime": "2019-08-24T14:15:22Z",

      "parameters": [

        {

          "id": "SSYS\_BP\_PUSSERVICE",

          "xvalue": ["AAABfA=="],

          "rvalue": 1,

          "pvalue": "TCVERIF",

          "dvalue": "TCVERIF",

          "status": [1],

          "obtvalue": "2019-08-24T14:13:45.865Z"

        },

        …

      ]

    },

    …

}

## Unavailability Slots

### Introduction

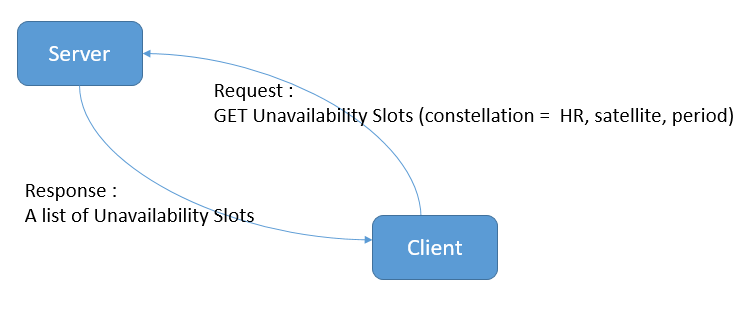
The "**Unavailability Slots**" Interface allows clients to straightforwardly retrieve a list of unavailability slots (slots unavailable for the mission) and that for a given constellation, satellite and period.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **Unavailability Slots**" request with constellation, satellite and period as parameters . In response, in case of success, the server returns a list of unavailability slots over the period given as input to the request.

[Unavailability\_Slots.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/307396983/Unavailability_Slots.pptx?api=v2&modificationDate=1678716284000&version=2)



Client/servers example :

|  |  |
| --- | --- |
| Clients | Server |
| MPS | SCRMS |
| SCRMS | FDS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **Unavailability Slots** request | Ask for a list of unavailability slots for a given constellation, satellite and period. | Query Parameters:   * constellation = HR * satellite * period |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **Unavailability Slots** response | 200 | Successfully returned a list of unavailability slots | List of unavailability slots :  {     constellation     satellite     unavailability period     unavailability type  }  … |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| constellation | String | Satellite constellation name |
| period | Period | Start date, end date |
| satellite | String | Satellite name |
| unavailability period | Period | Start date, end date (time slots of unavailability over the period given as input to the request) |
| unavailability type | Enum of strings | Unavailability type (reason of unavailability, for example : maintenance reason). |

## User Request Activation Request

### Introduction

The "**User Request Activation Request**" Interface allows clients to straightforwardly activate a "User Request".

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "POST **User Request Activation Request**" request with the "User Request" as parameter.

A "User Request" can be :

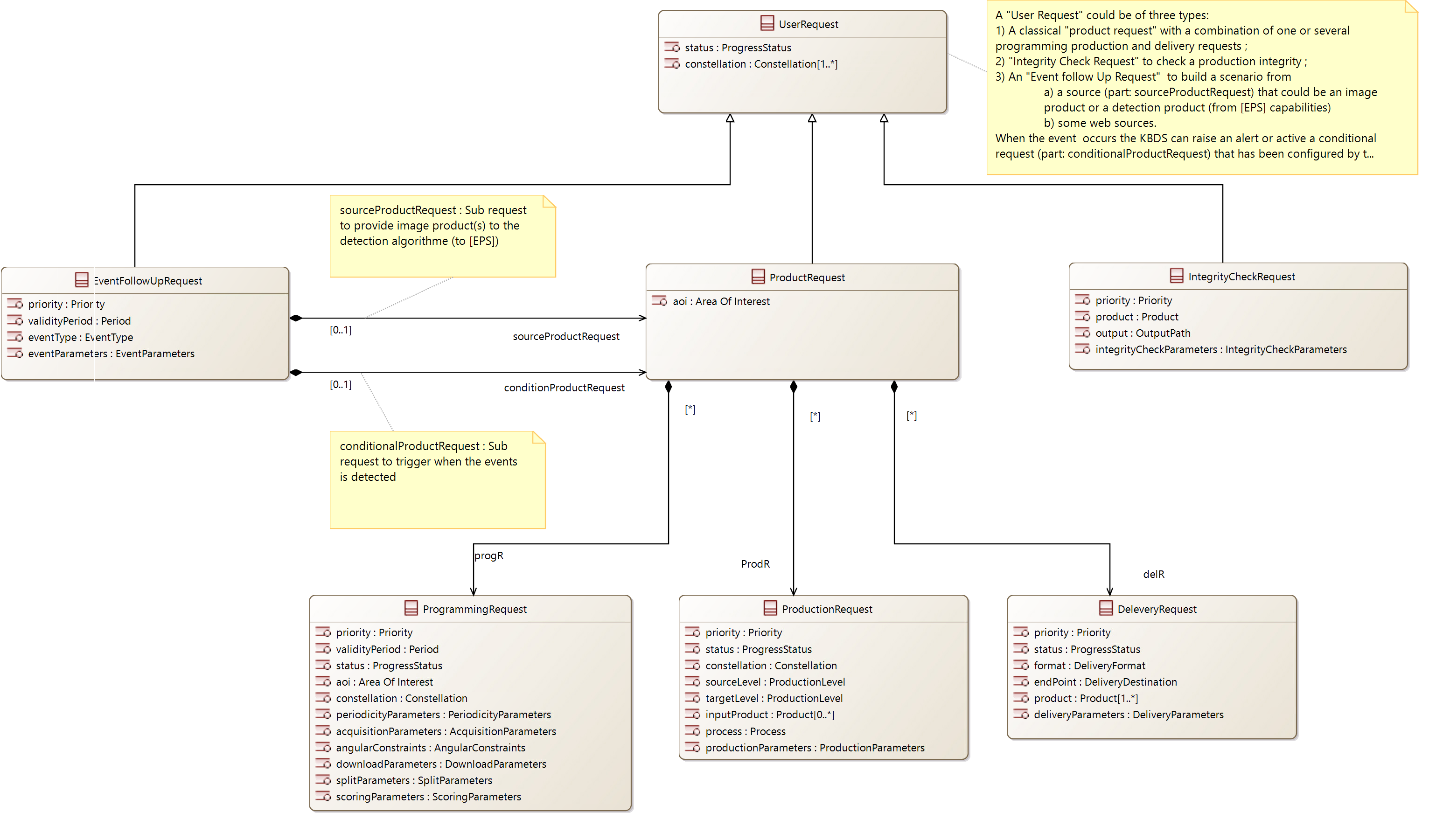
* A "Product Request"
* An "Integrity Check Request"" to check a product integrity (to DITS). Short Name : IntR
* An "Event Follow up Request" to request an event monitoring either web sourced or image sourced (to KBDS)). Short Name : EventFR

A "Product Request" is composed of :

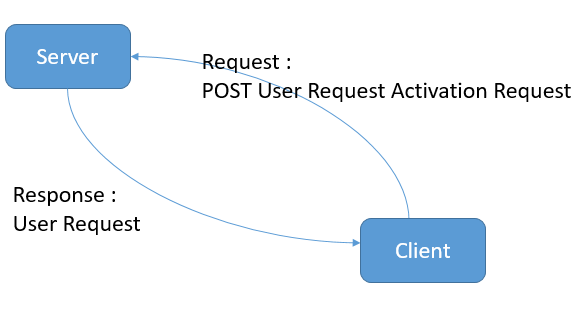
* 0 to N "Programming Request" to program new acquisition on a specific system (to MPS or IS). Short Name : ProgR
* 0 or N "Production Request" to request for image processing or reprocessing (to PPS or APS or EPS). Short Name : ProdR
* 0 or N "Delivery Request" to request for product Delivery, can contain tracability and integrity constraints (to DDS). Short Name : DelR

When an event occurs, [KBDS] raises an alert or active a conditional request (cf. conditionalProductRequest in the diagram class) that has been configured by the end user.

See  the below "User Request" class diagram :



The exchange between the client and the server is synchronous.



[User Request ACtivation request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308906711/User%20Request%20ACtivation%20request.pptx?api=v2&modificationDate=1679665299000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Clients | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| POST **User Request Activation Request**request | Ask to activate a "User Request" (cf. above definition). | UserRequest (see class diagram above) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| POST **User Request Activation Request**response | 200 | Successfully returned GUID which isthe unique identifier of the post | User Request  {  GUID,  ... (see User Request above)  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| acquisition parameters | AcquisitionParameters | Acquisition parameters:  Parameters to define the mode (monoscopic, stereo, etc.) and the constraints on the acquisition |
| angular contraints | AngularConstraints | Angular contraints :  Parameters to define the angular constraints of the acquisition |
| aoi | Area Of Interest | Area Of Interest |
| constellation | String | Satellite constellation name |
| delivery format | Enum of strings | Product delivery format |
| delivery parameters | DeliveryParameters | Delivery parameters : Delivery as products arise or Delivery at the end of all the productions, automatic Delivery, technical parameters (compression, croping, integrity, tec.) |
| download parameters | DownloadParameters | Download Parameters: Used to constraint the download of the acquisition including which ground stations can be used |
| end point | String | Where the product is delivered |
| event parameters | EventParameters | Event parameters as basic parameters (user id, alert, id, request id, detection type, ...) , alert parameter detection unit, detection product id, ...) or spatial and temporal parameters (location bbox, alert date) |
| event type | Enum of strings | Type of event the user wants to follow : Anomaly, Change, Feature on product or Web-event, etc. |
| guid | Integer | Globally Unique Identifier |
| integrity check parameters | IntegrityCheckParameters | Integrity Check Parameters :  can contain additional parameters need for integrity check processing |
| input product | InputProduct | Product to process |
| output product | String | Product path |
| periodic parameters | PeriodicParameters | Periodic Parameters :  Used to create a periodic programming request with period and minimum delay between the revisits (optional) |
| priority | Enum of strings | Request priority |
| process | String | Process applied to the product |
| product | Product | Product on which the integrity check relates or data produced by PPS, APS, EPS, etc. |
| production level | Enum of strings | Product production level |
| production parameters | ProductionParameters | Production Parameters : configuration parameters of the process (ie. as IQP or spectral bands to process) |
| source level | Enum of strings | Product source Level |
| splitParameters | SplitParameters | Split Parameters :  Parameters to specify how the mission should split the aoi such as the max mesh length |
| scoringParameters | ScoringParameters | Scoring Parameters:  Paramters to compute the score of the programming request such as weather |
| status | ProgressStatus | Composed of a status name and the cause of the status (in particular cause in case of rejection or error) |
| validity period | Period | Validity period of the request |

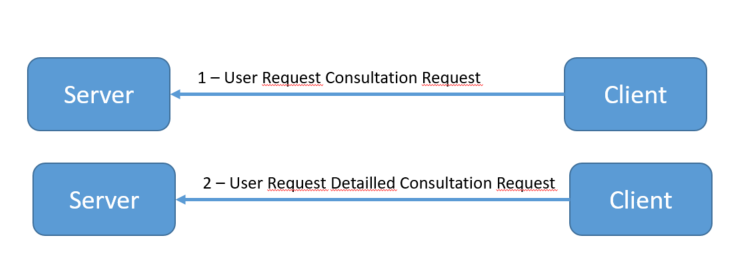
## User Request Consultation Request

### Introduction

The "**User Request Consultation Request**" Interface allows clients to consult a list of user requests according to search criteria. Requests presented to the client include the requested parameters and the "User Request GUID" is always sent.

This interface is strongly related to **User Request Detailed Consultation Request** interface**.** These two requests follow OGC api principles :

* 1a. [Client] makes a request with search criteria (attribute, logical operator, value) and a list of expected attributes to return
* 1b. [Server] returns the list of user requests with the attributes requested (the GUID is always returned) corresponding to the search criteria
* 2a. [Client] ask for the details (all the parameters) of a user request from the previous list using its GUID
* 2b. [Server] returns the detailed "User Request"

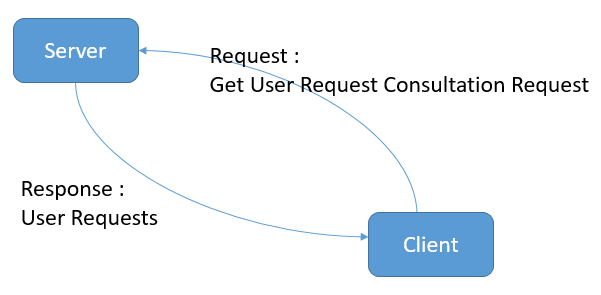


This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **User Request Consultation Request**" request with search criteria as parameters and a list of expected attributes.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/308906968/User%20Request%20ACtivation%20request.pptx?api=v2&modificationDate=1679666014000&version=1)

[User Request Consultation Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308906968/User%20Request%20Consultation%20Request.pptx?api=v2&modificationDate=1679672643000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **User Request Consultation Request**request | Ask to consult a list of user requests according to search criteria and a list of expected attributes | Query Parameters:   browsing criteria (attribute | logical operator | values) → including GUID, expected attributes (in response, limited number) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **User Request Consultation Request**response | 200 | In case of success return a list of "User Request"  ( each user request contains its GUID) | [  user request {  GUID,  ....  },  ...  ] |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| browsing criteria | BrowsingCriteria | Browsing critera (attributes , logical operator, values, etc.) |
| user request | UserRequest | User Request (see [User Request Activation Request](#scroll-bookmark-54) Interface) |

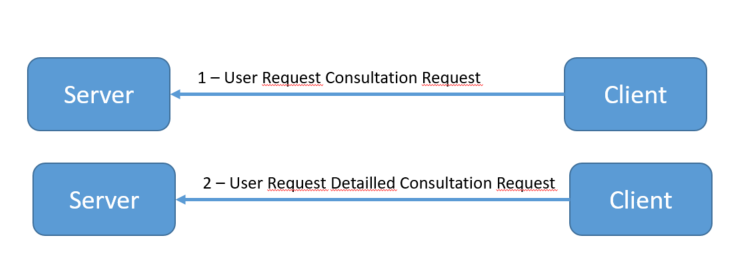
## User Request Detailed Consultation Request

### Introduction

The "**User Request Detailed Consultation Request**" Interface allows clients to ask for details of a "User Request". This "User Request" is identified by its GUID (User Request GUID is the parameter of the request).

This interface is strongly related to **User Request Consultation Request** interface**.** These two requests follow OGC api principles :

* 1a. [Client] makes a request with search criteria (attribute, logical operator, value) and a list of expected attributes to return
* 1b. [Server] returns the list of user requests with the attributes requested (the GUID is always returned) corresponding to the search criteria
* 2a. [Client] ask for the details (all the parameters) of a user request from the previous list using its GUID
* 2b. [Server] returns the detailed "User Request"

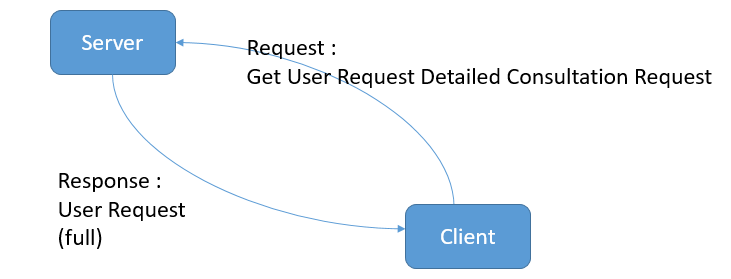


This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "GET **User Request Detailed Consultation Request**" request with  a GUID User Request as parameter

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907066/User%20Request%20Detailed%20Consultation%20Request.pptx?api=v2&modificationDate=1680017684000&version=2)

[User Request Detailed Consultation Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907066/User%20Request%20Detailed%20Consultation%20Request.pptx?api=v2&modificationDate=1680017684000&version=2)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **Request** |
| GET **User Request Detailed Consultation Request**request | Ask for the details (all attributes) of a user request from the previous list using its guid | Query Parameters:   * user request guid |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **User Request Detailed Consultation Request**response | 200 | In case of success the request return a full  "User Request" | user request {  GUID,  ....  } |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| user request guid | Integer | Globally Unique Identifier |
| user request | UserRequest | User Request (see [User Request Activation Request](#scroll-bookmark-54) Interface) |

## User Request Update Request

### Introduction

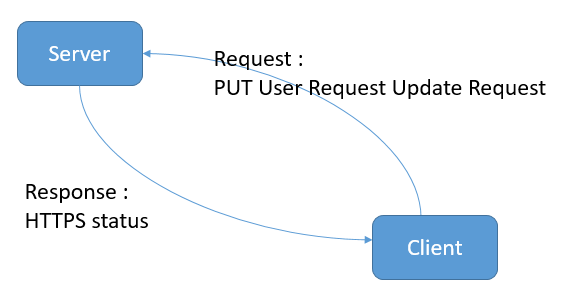
The "**User Request Update Request**" Interface allows clients to update a User Request.

This Web Service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

### Dynamics of exchanges

The client sends a HTTPS/REST "PUT **User Request Update Request**" request with GUID User Request and User Request updates as parameters.

The exchange between the client and the server is synchronous.

[](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907122/User%20Request%20Consultation%20Request.pptx?api=v2&modificationDate=1679672694000&version=1)

[User Request Update Request.pptx](https://wiki-external.thalesaleniaspace.fr/download/attachments/308907122/User%20Request%20Update%20Request.pptx?api=v2&modificationDate=1679673336000&version=1)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| UAS | FS |

### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| PUT **User Request Update Request**request | Ask to update a User Request | {  user request  {       user request GUID       ... (updates : see [User Request Activation Request](https://wiki-external.thalesaleniaspace.fr/display/DEOP/User+Request+Activation+Request?src=contextnavpagetreemode) )        }  } |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| PUT **User Request Update Request**response | 204 | In case of success return HTTPS status | HTTPS Status 204 |

Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| user request | UserRequest | User Request (see [User Request Activation Request](#scroll-bookmark-54) Interface) |

# Standard interfaces

## CCSDS TC SPACE DATA LINK PROTOCOL (232.0-B-3)

### Introduction

The purpose of this document is to specify a Telecommand (TC) protocol to use for ground-to-space or space-to-space communications.

This document is a publication of the Consultative Committee for Space Data Systems (CCSDS).

### Applicable documents

TC SPACE DATA LINK PROTOCOL (CCSDS 232.0-B-3) is available at: <https://public.ccsds.org/Pubs/232x0b3s.pdf>

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use this interface.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| FOS | CS | Clear TC |

## CCSDS TM SPACE DATA LINK PROTOCOL (132.0-B-2)

### Introduction

The purpose of this document is to specify a Telemetry (TM) protocol to use for ground-to-space or space-to-space communications.

This document is a publication of the Consultative Committee for Space Data Systems (CCSDS).

### Applicable documents

TM SPACE DATA LINK PROTOCOL (132.0-B-2) is available at: <https://public.ccsds.org/Pubs/132x0b2s.pdf>

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use this interface.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| CS | FOS | Clear TM |

## ESA-AUXIP V1.2 interfaces

### Introduction

The purpose of this document is to specify https RESTful Application Programming Interfaces (APIs) through which Auxiliary data may be discovered and downloaded by authorised users from Auxiliary Interface delivery Points (AUXIP). Each AUXIP represents a standard pick-up point for all Sentinel Auxiliary data collected or generated by the Auxiliary Data Gathering Service, the POD Service and the Cal/Val Service, thus replacing the diverse interfaces currently in place inside each Sentinel PDGS.  
This specification uses the latest OData v4 standard for REST-ful APIs. The API interface is intended to be compliant with the OData v4 standard, however, it is not necessary to fully implement the OData v4 standard. Compliance to the ICD can be considered as an API contract with a set of API OData v4 style calls, which have to be supported, that shall follow OData query conventions such as URL syntax, filter syntax, content accessing conventions, etc.

### Applicable documents

Cf.   [[RD-](#scroll-bookmark-278)7] : ESA-EOPG-EOPGC-IF-10 / Auxiliary Data Interface Delivery Point Specification (ESA-AUXIP)

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the AUXIP API.

* *Example : PPS-ADGS  :  On demand, External Image Auxiliary Data (Atmospheric data, Ground weather forecast data, Earth rotation and frame reference info, etc.) are retrieved from ADGS (AUXIP-Server) by PPS (AUXIP-client). The ESA-AUXIP standard interface reponds to Pickup Point requierements  :*
  + *Asynchronous Exchange*
  + *Centralized data*
  + *One Pickup-Point on each data producer*
  + *Choice of data according to the characteristics of the data*
  + *Data described as a file*
  + *Etc.*

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| PPS/APS | ADGS | External IAD |
| PPS/APS | IQS | Internal IAD |
| MPS | ADGS | Auxiliary Data |

The following hereunder sections present the format of the different files that can be retrieved from these interfaces.

### Climato

The "**Climato**" are retrieved through an ESA-AUXIP interface (see [ESA-AUXIP V1.2 interfaces](#scroll-bookmark-278)).

This page does not redefine the interface dynamics but only the format of the Climato file returned by the pickup point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned the requested file | Zipped .grib2 file |

### Data Elevation Model

### GIPP

The "**GIPP**" are retrieved through an ESA-AUXIP interface (see [ESA-AUXIP V1.2 interfaces](#scroll-bookmark-278)).

This page does not redefine the interface dynamics but only the format of the GIPP file returned by the pickup point.

These files are mission specific, so no generic format is defined but JSON is recommanded.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned a file | Mission specific format. JSON recommanded |

### Ground Reference Images

### IERS

The "**IERS**" are retrieved through an ESA-AUXIP interface (see [ESA-AUXIP V1.2 interfaces](#scroll-bookmark-278)).

This page does not redefine the interface dynamics but only the format of the IERS file returned by the pickup point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned the requested file | Format IERS ASCII standard  example : |

### Weather Forecasts

The "**Weather Forecasts**" are retrieved through an ESA-AUXIP interface (see [ESA-AUXIP V1.2 interfaces](#scroll-bookmark-278)).

This page does not redefine the interface dynamics but only the format of the weather forecast file returned by the pickup point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Status code** | **Description** | **JSON Responses** |
| GET **File** response | 200 | Successfully returned the requested file | Zipped .grib2 file |

## ESA-LTAIP V1.7 interfaces

### Introduction

The  purpose  of  this  document is  to  specify  the  interfaces  of  the  Long  Term  Archive (LTA) within  the Copernicus  Space  Component (CSC) Ground  Segment(GS) as  part  of  the overall  CSCESA Operations Framework. In particular this document aims tospecify an https RESTful Application Programming Interface (API) of the LTAthrough which Copernicus Sentinel data products may be queried anddownloaded by authorised users. This specification uses the latest OData v4 standard for RESTful. The API interface is intended to be compliant with  the  OData  v4  standard,  however,  it  is  not  necessary  to  fully  implement  the  OData  v4  standard.  Compliance to the ICD can be considered as an API contract with a set of API OData v4 style calls, which have  to be supported, that shall follow OData query conventions such as URL syntax,  filter syntax, content accessing conventions, etc

### Applicable documents

Cf.   [[RD-](#scroll-bookmark-295)4] : ESA-EOPG-EOPGC-IF-2 / Copernicus Space Component Long Term Archive Interface Control Document (ESA-LTAIP)

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the LTAIP API.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| PPS | ACS | Reprocessing products retrieval |
| EPS | ACS | Enhanced processing inputs retrieval |
| APS | ACS | Advanced processing inputs retrieval |

## ESA-PRIP V1.7 interfaces

### Introduction

The purpose of this document is to specify an https RESTful Application Programming Interface (API) through  which Sentinel (or Auxiliary) data products may be discovered and downloaded by authorised users from a Production Interface delivery Point (PRIP).

This specification uses the latest OData v4 standard for REST-full APIs. The API interface is intended to be compliant with the OData v4 standard, however, it is not necessary to fully implement the OData v4 standard.

Compliance to the ICD can be considered as an API contract with a set of API OData v4 style calls, which have to be supported, that shall follow OData query conventions such as URL syntax, filter syntax, content accessing conventions, etc.

### Applicable documents

Cf.   [[RD-5](#scroll-bookmark-300)] : ESA-EOPG-EOPGC-IF-3 / Production Interface Delivery Point Specification (PRIP)

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the PRIP API.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| ACS | PPS | Archiving products retrieval |
| ACS | EPS | Archiving products retrieval |
| ACS | APS | Archiving products retrieval |
| ACS | KBDS | Archiving products retrieval |
| ACS | INTS | External products retrieval |
| ACS | MAPQS | Archiving products retrieval |

## ESA-XBIP V1.1 interfaces

### Introduction

X-band Ground Station Data Delivery Point Interface (XBIP) defines the file format applicable to the satellite raw binary data files acquired at the X-band Ground Station as well as the structure of the data delivery point.

The data files are transferred in a well-structured directory tree, to be created in the data delivery point host by the X-band Ground Station service provider. The data delivery point contains a collection of files, i.e. xml descriptive files and raw binary data files containing the satellite generated CADUs.  
The data files are segmented in blocks of configurable dimension, i.e. CADU chunks, (in order to ease the operations of file handling and transfer), called DSDBs (Data Session Data Block).

The processing entity (user) is granted access to (selected) data delivery point directories by the acquisition service provider. The processing entity discovers the start of a new acquisition session for a given satellite by polling the data delivery point at the relevant root directory (for either NOMINAL or RETRANSFER sessions) and keeps track of the already successfully processed sessions in order to avoid attempting several times the download of a same session.

Upon discovery of a new session, the receiving entity can start triggering the retrievals by HTTPS of the corresponding DSDB files.

### Applicable documents

Cf.   [[RD-3](#scroll-bookmark-305)] : ESA-EOPG-EOPGM-SP-1 / X-band Ground Station ICD (XBIP)

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the XBIP API.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| PPS | Ground station | Raw telemetry |

## OGC API Processes V1.0.0 interfaces

### Introduction

The OGC API Processes standard supports the wrapping of computational tasks into executable processes that can be offered by a server through a Web API and be invoked by a client application. The standard specifies a processing interface to communicate over a RESTful protocol using JavaScript Object Notation (JSON) encodings. The standard leverages concepts from the OGC Web Processing Service (WPS) 2.0 Interface Standard but does not require implementation of a WPS. The Core part of the standard is called OGC API - Processes - Part 1: Core. The Core part of the standard supports the wrapping of computational tasks into executable processes that can be offered by a server through a Web API and be invoked by a client application either synchronously or asynchronously. The OGC API Processes Standard specifies a Web API that enables the execution of computing processes, the retrieval of metadata describing their purpose and functionality and the retrieval of the results of the process execution. The API makes use of different HTTP methods, namely GET, POST and DELETE.

### Applicable documents

Cf. [R[D-](https://wiki-external.thalesaleniaspace.fr/pages/createpage.action?fromPageId=297633768&linkCreation=true&spaceKey=DEOP&title=AD-1)7] : [OGC API](https://docs.ogc.org/is/18-062r2/18-062r2.htm)[processes](https://ogcapi.ogc.org/processes/), OGC 2023, <https://ogcapi.ogc.org/processes/> ;  <https://docs.ogc.org/is/18-062r2/18-062r2.html>

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the OGC API Processes.

|  |  |  |
| --- | --- | --- |
| Client | Server | DOMINO-X logical Interfaces |
| FS | PPS | Get Capabilities  PPS Reprocessing Request & Follow up |
| FS | KBDS | Get Capabilities  Event Follow-up Request & Follow up |
| FS | APS | Get Capabilities  APS Production Request & Follow up |
| FS | DDS | Get Capabilities  DeliveryRequest & Follow up |
| FS | EPS | Get Capabilities  EPS Production Request & Follow up |
| FS | DITS | Get Capabilities  ImageSignatureRequest & Follow up  ImageIntegrityVerificationRequest & Follow up |
| FS | INTS | Get Capabilities  External Product Retrieval Request  ProdR Follow-up |
| OMS | ALL | Maintenance Commands |

### Falsification Detection Request & follow-up

### Image Signature Request & follow-up

### PPS reprocessing request & follow-up

## STAC API V1.0.0 interfaces

### Introduction

At its core, the SpatioTemporal Asset Catalog (STAC) API specification provides a common structure for describing and cataloging spatiotemporal assets.

A *spatiotemporal asset* is any file that represents information about the earth captured in a certain space and time.

### Applicable documents

STAC API specification and interface is available at: <https://stacspec.org/en>

### DOMINO-X logical interfaces

The following DOMINO-X logical interfaces use the STAC interface:

|  |  |  |
| --- | --- | --- |
| ACS | UAS | Catalog structured search |
| ACS | UAS | Catalog natural language search |
| ACS | IQS | Catalog structured search |
| ACS | KBDS | Enhanced product metadata |
| ACS | MPS | Product metadata update |
| ACS | MAPQS | Product metadata update |
| ACS | IQS | Product metadata update |
| ACS | KBDS | Thematics catalog request |
| ACS | MAPQS | Catalog structured search |

STAC request using POST operation. Response is a STAC file.

The following DOMINO-X logical interfaces use the STAC interface extended with a HTTPS GET file download.

|  |  |  |
| --- | --- | --- |
| ACS | PPS | Reprocessing input retrieval |
| ACS | EPS | Enhanced processing input retrieval |
| ACS | APS | Advanced processing input retrieval |
| ACS | IQS | Product retrieval |
| ACS | DDS | Product retrieval |

First step: STAC request using POST operation. Response is a STAC file.

Second step: File request using GET operation. Response is a file compatible to Pivor Format Definition ([Work products](https://wiki-external.thalesaleniaspace.fr/display/DEOP/Work+products))

### Catalog Structured Search

### Product Metadata Update

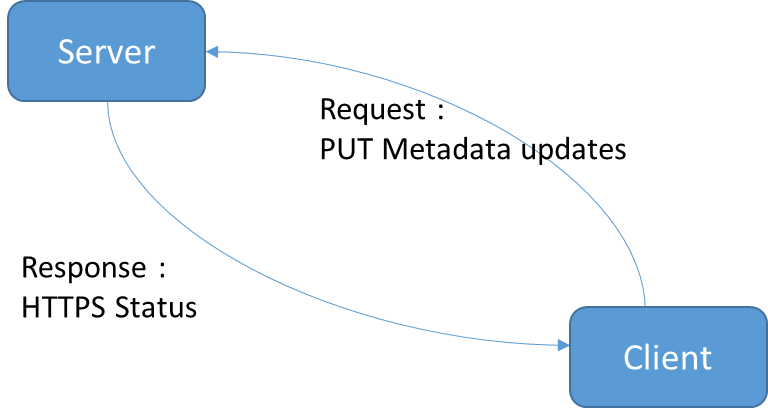
#### Introduction

The "**Product Metadata update**" Interface allows clients to update some metadata of a product.

This web service is based on OPENAPI 3.0 standard interface. The supported format for designing the API is JSON.

#### Dynamics of exchanges

The client sends a HTTPS/REST "PUT **Metadata updates**" request with a product id (or catalogue id) and the list of medata and associated value to update.



[Product\_Metadata\_update.pptx](#scroll-bookmark-328)

Client/server example :

|  |  |
| --- | --- |
| Client | Server |
| MPS | ACS |
| MAPQS | ACS |

#### Request/Response

|  |  |  |
| --- | --- | --- |
| Interface | **Description** | **JSON  request** |
| **PUT Product Metadata update**request | Send a list of medata and values to update for a given product | {     Catalogue ID,     [{metadataName, value}, …]  } |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Interface** | **Status code** | **Description** |
| **PUT Product Metadata update** response | 200 | Status code to confirm the update is ok |

*Note on status code :*

* *200 OK: if the request is accepted and a response can be returned*
* *400 Bad Request-401 Unauthorized: if the requesting client is unauthorised*
* *404 Not Found*
* *429 Too Many Requests: if a quota is exceeded*
* *500 Internal Server Error*

#### Data type

|  |  |  |
| --- | --- | --- |
| Data | Type | Description |
| Catalogue ID | String | Product ID or Catalogue ID known in the catalogue |
| medata name | String | Name of the metadata to update or add |
| value | String, integer, float, enum, etc. | New value of the medata, its type depends on the metadata |

### Product Retrieval