esac

European Space Astronomy Centre
P.O. Box 78
28691 Villanueva de la Cañada
Madrid
Spain
T +34 91 8131 100
F +34 91 8131 139
www.esa.int

DOCUMENT

Solar Orbiter Observation Timeline Export ICD

Prepared by **Thanos Tsounis, Chris Watson, David Williams**

Reference **SOL-SGS-ICD-0012**
Issue **0**
Revision **1**
Date of Issue **12/09/2019**
Status **Draft**
Document Type **ICD**
Distribution

APPROVAL

Title Solar Orbiter Observation Timeline Export ICD	
Issue 0	**Revision** 1
Author(s) Thanos Tsounis, Chris Watson, David Williams	**Date** 12/09/2019
Approved by	**Date**
SOC Development Manager	
EPD representative	
EUI representative	
MAG representative	
METIS representative	
PHI representative	
RPW representative	
SOLO-HI representative	
SPICE representative	
STIX representative	
SWA representative	

CHANGE LOG

Reason for change	Issue	Revision	Date
N/A	0	1	12/09/2019

CHANGE RECORD

Issue 0	Revision 1		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
N/A			

Table of contents:

1 INTRODUCTION

For a Solar-System mission, Solar Orbiter requires an unusually high degree of coordination between instruments. A major step in organising this coordination is the LTP planning meeting of the SOWG. The Observation Timeline Export (OTE) provides a record of this plan. It is complementary to the other planning files that are created from the LTP.

1.1 Overview of planning cycles WRT observation timeline export

Planning cycle	Application of OTE
SAP / Mission-level	*None*
LTP	*Major* Allows instrument teams to use software to dynamically check the in-work plan during the SOWG LTP meeting.
MTP	*Major* Same as for STP, below.
STP	*Major* Contributes part of the "documentary" planning information about the science intent of the planning period. Provides the SOOP-ID which is used by the ITs to create ObservationIDs within the IORs.
i-VSTP	*None*

1.2 What the LTP observation timeline export is for

The export provides an electronically readable record of an LTP within SOOP-kitchen. Generally, it is expected that humans will prefer to view an LTP plan via the web GUI provided by the server, but where software has to view an LTP plan this export is the provided mechanism.

The export also provides the link between the conceptual plan and the later implementation via the SOOP IDs (see section 1.6).

1.3 Recap of the SOOP-Kitchen LTP plan

Since the export is a representation of a SOOP Kitchen LTP plan, it is appropriate to recall how an LTP plan works.

- LTP establishes the necessary **coordination** between instruments such that subsequent MTP/STP planning can in principle be done by the instrument teams in isolation
- LTP is necessarily coarse-grained
  - Exactly how coarse-grained will have to be tuned with experience and may vary instrument-by-instrument.
  - On one hand, certain decisions are fixed at LTP

        - The number and positioning of pointing events is fixed
            - This means for example that ITs need to be sure that pointing dwell periods inside the LTP plan are adequate for their needs
    - On the other hand, instrument teams are expected to need to "work around" constraints later at the detailed IOR level to ensure that, e.g.,
        - EMC windows are respected by the instrument
        - Resource usage is respected by the instrument
        - Safety constraints are respected during platform events

      As such, the LTP export does not fully determine the low-level commanding that comes later in the IORs.
    - More generally, we can say that LTP is working in the sense of conceptual "modes" rather than attempting to represent atomic data acquisitions/images. And often we will choose to plan these "modes" through platform events that interfere because at LTP-level it is not worth breaking the mode over these interruptions.

1.4 Reference Documents

[GFTSICD] "Solar Orbiter File-Transfer SOC<->Instrument Teams ICD", Emilio Salazar & Chris Watson (ESAC) SOL-SGS-ICD-0009, v1.1, April 2018.

[IORICD] "Solar Orbiter Instrument Operation Request Interface Control Document (IOR ICD)", Christopher Watson (ESAC), SOL-SGS-ICD-0012, v1.1, March 2019

[METADATA] "Metadata Definition for Solar Orbiter Science Data", SOL-SGS-TN-0009, v2.4, September 2019

[SAP] "Solar Orbiter Science Activity Plan", SOL-EST-PL-8539, v0.1, July 2017

[SPKTUM] "SOOP Kitchen User Manual", https://issues.cosmos.esa.int/solarorbiterwiki/display/SOSP/SOOPKitchen+User+Manual

1.5 Timeline export in context

Figure 1 illustrates the context of the export.

During the LTP meeting, instrument teams may pull an export of the plan under construction from the GUI. This allows the possibility of a sanity check by software of the developing plan, for those teams that prefer this to (or in addition to) visual inspection.

Following the meeting a final version of the plan is pushed to the instrument teams via GFTS. It is only this final version of the plan, distributed by SOC, that is the applicable version from which instrument teams prepare their IORs for MTP/STP. All other (typically prior) versions of the plan are informative only, and are intended to help participants in the Long-Term Planning meeting to converge on the final plan.

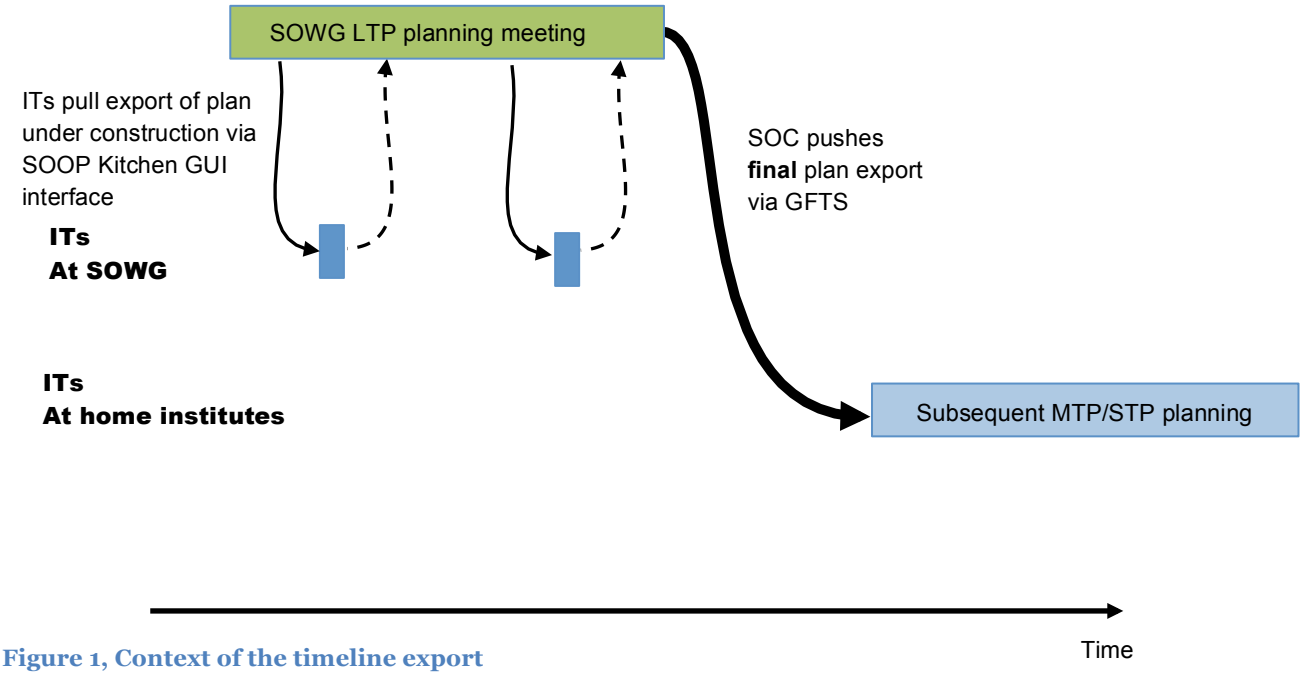

Figure 1, Context of the timeline export

1.6 SOOP IDs

The SOOP IDs are represented in the "socId" elements in the export. See [IORICD] for the internal structure of this element. Broadly, most component parts are base58, but there is an exception for zero: for example, "000" being used where an observation is not part of any SOOP.

1.7 Acronyms

BSUP	BoreSight Update *Short name given to the XML files that are used to achieve the update described in this ICD*
CP	Cruise Phase
EFECS	Enhanced Flight Events and Communication Skeleton *Also called planning skeleton. This is a SOC-extended version of the FECS that comes from MOC which details the spacecraft events*
EMP	Extended Mission Phase
FITS	Flexible Image Transport System
IOR	Instrument Operational Request
JSON	JavaScript Object Notation *A lightweight data-interchange format*
ITs	Instrument Teams
LIF	Low-latency Instrument Frame, as defined in [LLFITSICD]. *Not strictly a single frame, since each optical path or detector has its own LIF.* *One way to think of the LIF is as a Solar Orbiter* ***convention*** *on how the LL FITS spatial axes shall be used, such that all Solar Orbiter imaging instruments shall have e.g. CTYPE1 and CTYPE2 defined (in ordering and sign) in a way that they map to the sky in the same broad sense, even if the precise orientations are not identical at low-level.*
LL or LLD	Low-Latency Data *That "thin-slice" of science data that can always be downlinked promptly to ground*
LTP	Long-Term Planning
MOC	Mission Operations Centre. *For Solar Orbiter this is ESOC in Darmstadt.*
MTP	Medium-Term Planning
NAIF	Navigation and Ancillary Information Facility *The part of NASA responsible for maintaining the SPICE software toolkit used (broadly) for spacecraft geometry calculations. We use "NAIF-SPICE" to distinguish this toolkit from the SPICE instrument*
NECP	Near Earth Commissioning Phase
NMP	Nominal Mission Phase
OTE	Observation Timeline Export (subject of this ICD)
p-VSTP	That part of VSTP where pointing is planned
REST	Representational State Transfer
RS	Remote Sensing
RSW	Remote Sensing Window
SAP	Science Activity Plan *The top-level science plan of how solar orbiter will utilise the time within CP, NMP, EMP to fulfil its science goals*
SC	Spacecraft
SOC	Science Operations Centre *For Solar Orbiter this is ESAC near Madrid*
SOWG	Science Operations Working Group
STP	Short-Term Planning
TAC	Turn-Around Calibration *Complements LLD as prompt science link to ground. Because a "fatter slice" of science can come through this (compared to LLD) it is normally OFF, and only enabled for specific activities that need it (those where there is a mandatory tight space->ground->space loop needed), and then only when the downlink can support it.*
VSTP	Very Short Term Planning
WRT	With Respect To

2 DELIVERY MECHANISMS

There are two ways to obtain the observation timeline export. At the end of the planning exercise the SOWG-agreed, consolidated list of observation instances will be available through GFTS, along with other planning products. During the planning exercise, the export will be available through SOOP Kitchen via a REST API.

2.1 GFTS

There is one JSON-format OTE file per instrument made available though GFTS, containing the observation instances corresponding to that instrument. The format of the observation timeline export's filename, as distributed by GFTS to the instrument teams, is:

```
Sxxx_observation_timeline_export_Myy_Vzz.json
```

Where *Sxxx* is the 4-letter instrument code (e.g., "SEPD"), *yy* is the LTP number covered by the corresponding EFECS and *zz* is the version number of the corresponding EFECS. For more information please refer to [GFTSICD].

2.2 SOOP Kitchen GUI

For a given plan, it is possible to see a list of all its versions, and the changes made in each version with respect to the last. This interface is documented in the SOOP Kitchen User Manual [SPKTUM]. For each listed version there are multiple actions possible in the GUI, of which one is to "Export". The result of clicking a button marked 'Export' is that a ZIP archive of all the instrument timelines (and additional context files), can be downloaded. Within this archive is a subdirectory called 'PlanningOutput', in which resides a timeline file for each instrument, named:

```
Sxxx_observation_timeline_export.json
```

Note that this filename is missing the LTP and EFECS version number (see §2.1). This is because the planning outputs are not considered final until they have been distributed through GFTS, and the EFECS is not assigned a version number until all the events necessary for MTP/STP planning have been generated and included.

Downloading OTE files in this manner is somewhat cumbersome for interfacing with instrument teams' own software, however. As a result, the API can be called as described below (§2.3).

2.3 REST API

2.3.1 Using a browser window

The SoopKitchen server exposes a REST API using the HTTP GET operation, which allows a user to obtain the export on demand for any version of any plan. The operation requires

authentication. This can be easily achieved using a browser and typing in the appropriate URL. The user just needs to be logged in SoopKitchen using the same browser. Then the browser will send the authentication information with all subsequent requests. The base request URL is:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/<plan_name>
```

where <*plan_name*> is the name of the plan that is going to be exported. This request will export the latest version of the plan and it will include all instruments. Optionally the internal version to be exported can be specified:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/<plan_name>/<plan_version>
```

Furthermore, a list of instrument names may be given as a parameter to the above request, with or without the optional internal version being specified:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/<plan_name>/<plan_vesrion>?instruments=<comma_separated_instrument_names>
```

In this manner the export will only contain the observation and SOOP instances corresponding to the list of instruments provided. Both the parameter name and its value are case sensitive.

2.3.1.1 Examples

Export SWA and SoloHI of version 2 of plan LTP01_May2020-Dec2020:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/LTP01_May2020-Dec2020/2?instruments=SoloHI,SWA
```

Export SWA of the latest version of plan LTP01_May2020-Dec2020:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/LTP01_May2020-Dec2020?instruments=SWA
```

Export all instruments of the latest version of plan LTP01_May2020-Dec2020:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/LTP01_May2020-Dec2020
```

2.3.2 Using a programming/scripting language

The REST API for can also be used by a client written in a programming or scripting language for maximum automation. There is an authentication step needed prior to the HTTP GET request. This is an HTTP POST request on the following URL:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/
authentication?j_username=<username>&j_password=<password>&remember
-me=true&submit=Login
```

where <*username*> is the username of the user and <*password*> the corresponding password. The following headers must be set:

Header	Value	Description
"X-XSRF-TOKEN"	String	A random token used against cross-site request forgery attacks. It is usually generated by a library in the programming language that handles the HTTP requests.
"Cookie"	"XSRF-TOKEN=<X-XSRF-TOKEN>"	Set the value of the X-XSRF-TOKEN in a cookie named XSRF-TOKEN
"Connection"	"keep-alive"	Instructs the server to keep the user logged in

Then the response headers "JSESSIONID" and "remember-me" must be cached and used for all subsequent requests. For example, for getting the latest version of plan 'LTP01_May2020-Dec2020' we need to issue the following HTTP GET request:

```
https://solarorbiter.esac.esa.int/soopkitchen/api/plan/ote/LTP01_Ma
y2020-Dec2020
```

with the following HTTP headers:

Header	Value	Description
"X-XSRF-TOKEN"	String	The same random token used in the login request.
"remember-me"	String	Value obtained from the response headers of the login request
"Cookie"	"'XSRF-TOKEN=<X-XSRF-TOKEN>; JSESSIONID=<JSESSIONID>; remember-me=<remember-me>"	Sets three cookies: the XSRF-TOKEN with the same value as the X-XSRF-TOKEN and the JSESSIONID and remember-me with the same values as the response headers of same name of the login request.

2.3.2.1 Python Example

Below there is a sample python script that will authenticate the user and then request a JSON export of a specific plan and/or version. The username, password and plan name must be set in the script along with, optionally, the plan version.

```
import requests

SPKT_URL = "https://solarorbiter.esac.esa.int/soopkitchen/"
OTE_URL = SPKT_URL + "api/plan/ote/"
LOGIN_URL = SPKT_URL + "api/authentication?j_username=%s&j_password=%s&remember-
me=true&submit=Login"

# set username and password
username = ""
password = ""
# plan to export
plan_name = ""
# change if you want specific version
plan_version = ""

request_client = requests.session()
login_url = LOGIN_URL % (username, password)
request_client.get(login_url)

# sets cookie
xref_token = request_client.cookies['XSRF-TOKEN']
login_headers = {
    'Cookie': 'XSRF-TOKEN=%s;' % xref_token,
    'X-XSRF-TOKEN': '%s' % xref_token,
    'Connection': 'keep-alive'
}

post = requests.post(url=login_url, headers=login_headers)

if not post.ok:
    print("Wrong request. Response Code = %s , %s" % (post.status_code,
post.reason))
    if 400 <= post.status_code < 404:
        print("----------------------")
        print("User/Password not valid")
        print("----------------------")
    exit(-1)

remember_me = post.cookies['remember-me']
jsession_id = post.cookies['JSESSIONID']

json_url = OTE_URL + plan_name
if plan_version != "":
    json_url = json_url + '/' + plan_version

json_export_headers = {
    'Cookie': 'XSRF-TOKEN=%s; JSESSIONID=%s; remember-me=%s' % (xref_token,
jsession_id, remember_me),
    'X-XSRF-TOKEN': '%s' % xref_token,
```

```
    'remember-me': '%s' % remember_me
}

plan = request_client.get(url=json_url, headers=json_export_headers).json()

print(plan)
```

3 CONTENT

There are three main components of the observation timeline export. The first one holds generic information pertaining to the version of the plan being exported. The second one is the list of the SOOP instances in the plan and the third one is the list of the observation instances. Only the third one is available in the formal export that is distributed via GFTS. The list of observation and SOOP instances will only contain the ones corresponding to the instrument(s) that have been selected though the REST API. If no instrument is selected then all SOOP and observation instances are available. Through GFTS the observation instances are already split into different files, one for each instrument.

3.1 Info

This component contains generic information on the specific version of the plan that is exported.

3.1.1 Attribute Table

Attribute	Type	Description
name	String	The name of the plan
planId	String	The ID of the plan
internalVersion	Positive Integer	The incremental internal version of the plan
baselineVersion	Non-negative Integer	The baseline version of the plan
lastUpdated	String	The timestamp of the internal version of this plan in `yyyy-MM-dd'T'HH:mm:ss'Z'` format
creationVersion	String	The SoopKitchen version used to create this plan
exportVersion	String	The SoopKitchen version used for this export
observationDefinitions	String	The name of the observation definitions of this plan
eventCollections	String	The name of the event collection of this plan
eventSchemaName	String	The xml event file schema definition name
eventSchemaVersion	String	The xml event file schema definition version
type	String	Whether it's an operational plan, a child plan, o SOOP template or a floating timeline

3.1.2 Example

```
"info": {
  "name": "LTP01_May2020-Dec2020",
  "planId": "010A",
  "internalVersion": 579,
  "baselineVersion": 0,
  "lastUpdated": "2019-07-25T14:23:48Z",
  "creationVersion": "7.1.1",
  "exportVersion": "7.1.1",
  "observationDefinitions": "Nexus_v7.1.0",
  "eventCollections": "LTP01_FECS02_PTEL00002",
  "eventSchemaName": "e-fecs",
  "eventSchemaVersion": "2.0",
  "type": "Operational Plan"
}
```

3.2 SOOPs

This component contains a list of the SOOP instances within the exported plan.

3.2.1 Attribute Table

Attribute	Type	Description
startDate	String	The start date of the first observation in the SOOP instance in `yyyy-MM-dd'T'HH:mm:ss'Z'` format
endDate	String	The end date of the last observation in the SOOP instance in `yyyy-MM-dd'T'HH:mm:ss'Z'` format
encodedSoopType	String	The three letter ID of the SOOP type
soopInstanceId	String	The SOOP instance ID
soopType	String	The SOOP type

The [SAP] is the source of the SOOP Type values, which are intended to be scientifically meaningful. That document is expected to evolve substantially in the early part of the mission, and it is therefore considered counterproductive to list the possible values of *soopType* in this document. However, note that the value(s) of *soopType* is/are equivalent to that of the SOOPNAME keyword as described in [METADATA]; similarly, *encodedSoopType* corresponds to SOOPTYPE (same document).

3.2.2 Example

```
"soops": [
  {
    "startDate": "2020-05-14T00:00:00Z",
```

```
    "endDate": "2020-06-03T10:17:00Z",
    "encodedSoopType": "IDF",
    "soopInstanceId": "111",
    "soopType": "I_DEFAULT"
  },
  {
    "startDate": "2020-06-18T04:00:00Z",
    "endDate": "2020-06-18T05:00:00Z",
    "encodedSoopType": "CC1",
    "soopInstanceId": "113",
    "soopType": "COORD_CALIBRATION"
  }
]
```


3.3 Observation instances

This component consists of a list of the observation instances within the exported plan. The structure will be similar for all observation instances, but the values of each instance's attributes (including whether they are finite or null) will depend on how a given instrument is modelled by SOC in SOOP Kitchen. The model is developed in consultation with each instrument team, so that the content (and any change therein) is mutually understood.

3.3.1 Attribute Table

Attribute	Type	Description
experiment	String	The instrument name
module	String	The module name
name	String	The observation name
startDate	String	The start date of the observation in `yyyy-MM-dd'T'HH:mm:ss'Z'` format
endDate	String	The end date of the observation in `yyyy-MM-dd'T'HH:mm:ss'Z'` format
duration	Positive integer	The duration of the observation in seconds
comment	String	A comment pertaining to the observation. It may be empty.
numberParameters	Object	A JSON object containing all the number parameters (parameters that have a numerical value) of the observation. Each attribute of the object is the name of the corresponding parameter and has another JSON object as value as per the table in section Number parameters. It may be empty.

listParameters	Object	A JSON object containing all the list parameters (parameters that have a string value from an enumerated list of possible values) of the observation. Each attribute of the object is the name of the corresponding parameter and has a unique string value. It may be empty.
power	Object	A JSON object as per the table in section Power.
flush	Object	A JSON object as per the table in section Flush. It may be null.
flows	Object	A JSON object containing all the data flows generated by the observation. Each attribute of the object is the name of the corresponding flow and has another json object as value as per the table in section Flows. It may be empty.
volumes	Object	A JSON object containing all the volumes of data produced by the observation. Each attribute of the object is the name of the corresponding data volume and has another json object as value as per the table in section Volumes. Each of the data flows of the observation will correspond to exactly one data volume, but there might be more. It may be empty.
compositeId	Object	A JSON object containing a detailed breakdown of the observations SOC ids as per the table in section Composite ID.
socIds	List	A list of strings with all the SOC ids that the observation instance has.

3.3.1.1 Number parameters

Attribute	Type	Description
value	Number	The numerical value of the parameter
unit	String	The unit of the value. It may be null.

3.3.1.2 Power

Attribute	Type	Description
value	Number	The numerical value of the power used by the observation
unit	String	The unit of the value. It may be null.

3.3.1.3 Flush

Only some instruments (*e.g.*, EUI, PHI) will be modelled as having flushes of data to the SSMM bulk-science store. Those that are modelled as such will have non-null values in this section. Flushes are not used in modelling data to LL or HK stores in the SSMM.

Attribute	Type	Description
value	Number	The numerical value of the data volume being flush from the internal memory to the SSMM.
unit	String	The unit of the value. It may be null.

3.3.1.4 Flows

Typically, those instruments which do not flush data to the SSMM bulk-science store will have sustained rates of data *flow* to those stores instead. Flows of type "LL" represent data passing to the Low-Latency store for all instruments. Flows of type "HK" are instrument housekeeping data passing to the SSMM (not OMM) housekeeping store for all instruments. Flows of type SCI are bulk science.

Attribute	Type	Description
value	Number	The numerical value of the data rate for the specific flow.
unit	String	The unit of the value. It may be null.
type	String	The type of the flow, e.g. HK, LL, SCI, etc.

3.3.1.5 Volumes

This attribute is informative: it either corresponds to the volume of data flushed, or it expresses the accumulated volume sent to a store over the observation instance's duration. In either case, the value of flush volume (§3.3.1.3) or of flow rate (§3.3.1.4) is the reference value to be used. The value represented here is merely a derived parameter.

Attribute	Type	Description
value	Number	The numerical value of the data volume.
unit	String	The unit of the value. It may be null.

3.3.1.6 Composite ID

Attribute	Type	Description
obsType	String	A four-character, case-sensitive, alphanumeric code for the observation type.
obsInstanceId	String	A three-character, case-sensitive, alphanumeric code for the observation instance.
compositeSoopIds	List	A list of JSON objects as per the table in section Composite SOOP ID.

3.3.1.7 Composite SOOP ID

Attribute	Type	Description
encodedSoopType	String	A three-character, case-sensitive, alphanumeric code

		for the SOOP type. The code "000" (three zeros) denotes that the observation exists outside a SOOP, without precluding that it is also associated with SOOP instance(s).
soopInstanceId	String	A three-character, case-sensitive, alphanumeric code for the SOOP instance ID. The code "000" denotes association with SOOP type "000".

3.3.2 Example


```
"observations": [
  {
    "name": "SWA_DPU_ON",
    "startDate": "2020-05-13T23:58:50Z",
    "endDate": "2021-01-01T00:00:00Z",
    "module": "DPU",
    "experiment": "SWA",
    "duration": 20044870,
    "numberParameters": {},
    "listParameters": {},
    "power": {
      "value": 9.6,
      "unit": "W"
    },
    "flush": null,
    "flows": {
      "HKFlow": {
        "value": 100.0,
        "unit": "b/s",
        "type": "HK"
      }
    },
    "volumes": {
      "HKFlow": {
        "value": 238.95347208423001,
        "unit": "MiB"
      },
      "SWA_TAV": {
        "value": 238.95347208423001,
        "unit": "MiB"
      }
    },
    "comment": "",
    "compositeId": {
      "obsType": "don2",
```

```
      "obsInstanceId": "111",
      "compositeSoopIds": [
        {
          "encodedSoopType": "000",
          "soopInstanceId": "000"
        }
      ]
    },
    "socIds": [
      "SSWA_010A_000_000_don2_111"
    ]
  },
  {
    "name": "RPW_HIGH1_COLD",
    "startDate": "2020-05-21T00:00:00Z",
    "endDate": "2020-05-22T04:33:00Z",
    "module": "COMMON_MODULE",
    "experiment": "RPW",
    "duration": 102780,
    "numberParameters": {
      "BURST_MINS": {
        "value": 240.0,
        "unit": "mins/day"
      },
      "SELECTIVE_FACTOR": {
        "value": 10.0,
        "unit": null
      }
    },
    "listParameters": {
      "COMPRESSION": "ON"
    },
    "power": {
      "value": 19.1,
      "unit": "W"
    },
    "flush": {
      "value": 20.1,
      "unit": "Mibyte"
    },
    "flows": {
      "RPW_NORMAL": {
        "value": 28054.166666666668,
        "unit": "b/s",
        "type": "SCI"
      },
```

```
      "RPW_BURST": {
        "value": 4045.0,
        "unit": "b/s",
        "type": "SCI"
      },
      "RPW_SELECTIVE": {
        "value": 4790.0,
        "unit": "b/s",
        "type": "SCI"
      },
      "HKFlow": {
        "value": 183.0,
        "unit": "b/s",
        "type": "HK"
      },
      "LLFlow": {
        "value": 93.0,
        "unit": "b/s",
        "type": "LL"
      }
    },
    "volumes": {
      "RPW_NORMAL": {
        "value": 343.72893105335254,
        "unit": "MiB"
      },
      "RPW_BURST": {
        "value": 49.56067819197901,
        "unit": "MiB"
      },
      "RPW_SELECTIVE": {
        "value": 58.68866465749801,
        "unit": "MiB"
      },
      "HKFlow": {
        "value": 2.2421765411946004,
        "unit": "MiB"
      },
      "LLFlow": {
        "value": 1.1394667668366,
        "unit": "MiB"
      },
      "RPW_TAV": {
        "value": 455.3599172108608,
        "unit": "MiB"
      }
```

```
    },
    "comment": "",
    "compositeId": {
      "obsType": "DfPp",
      "obsInstanceId": "114",
      "compositeSoopIds": [
        {
          "encodedSoopType": "IDF",
          "soopInstanceId": "111"
        },{
          "encodedSoopType": "IDA",
          "soopInstanceId": "113"
        },{
          "encodedSoopType": "000",
          "soopInstanceId": "000"
        },

      ]
    },
    "socIds": [
      "SRPW_010A_IDF_111_DfPp_114",
      "SRPW_010A_IDA_113_DfPp_114",
      "SRPW_010A_000_000_DfPp_114"
    ]
  }
]
```