EuroPlaNet-RI /
EuroPlaNet-Table Access Protocol

Draft 0.40e

EPN/JRA4-IDIS/Task 2

Editors:

Stéphane Erard (LESIA / OV-Paris), Pierre Le Sidaner (DIO / OV-Paris), Baptiste Cecconi

Contributors:

Pierre Le Sidaner, Stéphane Erard, Jérôme Berthier, Florence Henry, Marco Molinaro, Marco Giardino, Natacha Bourrel, Nicolas André, Michel Gangloff, Christian Jacquey, Markus Demleitner (or keep him for v2?)

Abstract

The goal of this document is to describe a protocol to access and retrieve Planetary Science data in general. This protocol will allow the user to select a subset of data from an archive in a standard way. This document describes the requirements to allow data providers to implement an EPN-TAP compliant service, based on the IVOA Table Access Protocol (TAP) specifications [RD6] and the EuroPlaNet Data Model [RD5].

Status of this document

This document is a development on the previous version 0.29 which was approved by the IDIS partners in June 2012. This is the last version 1 of the EPN-TAP protocol, as described in [RD25]. The protocol later evolved toward version 2, which solves several residual inconsistencies.

Acknowledgements

This work has been conducted in the frame of Europlanet-RI JRA4 work package.
The EuroPlaNet-RI project is funded by the European Commission under the 7th Framework Program, grant #228319 "Capacities Specific Programme".

Document change record

Issue	Date	Description of the modification	Contributions
Draft 0.1	15/07/2010	First Draft	P. Le Sidaner
0.11	11/01/10	Evolution	P. Le Sidaner
0.12	28/11/2011	Description of parameters	B. Cecconi, S. Erard, P. Le Sidaner
0.13	12/12/2011	Modification on many parameters	All VOPDC planetology
0.14	4/01/2012	Correction from Baptiste	B. Cecconi
0.15	7/1/2012	Corrections from Stéphane, integration of comments from recent meetings	S. Erard
0.16	20/01/2012	Merging all comments (SE, BC, MG)	P. Le Sidaner
0.17	9/02/2012	General update	S. Erard, P. Le Sidaner
0.18	13/02/2012	Reworked Section 2 + Added Appendices + reformatting + complete check & new comments throughout	S. Erard
0.19	13/02/2012	Comparison with ObsCore doc, extra comments	S. Erard
0.20	17/02/2012	Integrated comments	SE, JB, JCM, BC
0.21	25/02/2012	Further comments included Previous Appendix B => separate doc Completed service output, added UCDs Developed section 2	SE, PLS, BC
0.22	27/02	Evolution on time, spectral and spatial axis Changed idis (tap & core) to epn for Europlanet	BC, N. Andre, N. Bourel, C. Jacquey, M. Gangloff, PLS
0.23	22-27/03	Improvements, nasty details + consistency check	PLS, SE
0.24	6/06	Add 3 new parameters (angles)	SE, PLS
0.25	9/06	Better definition of “best practice”, introduction of Species	SE, PLS
0.26	11/06	Clarified calibration levels, answered some questions. Added element_name, Time_origin	SE, PLS
0.27	3/07	Integration of comments from IDIS general meeting & after implementation of first services. Included appendix C with definition file. Checked UCDs. Reworked figures.	SE, PLS, Sandrine Vinatier, Florence Henry
0.28	13/07	Discussions	SE, PLS
0.29	27/08	Minor corrections after implementation	PLS, SE
0.30	12/12	Complements from discussions & implementations: + string parameters in lower case + better definition of “best practice” + clarified “main product” issues + Reorganized optional parameters, introduced parameter attributes (TBC) + reorganized section 4/5, service properties now clearly identified	SE, with inputs from FH, MM, MG, PLS, BC and first services
0.31	21/1/2013	Corrections	SE, PLS, BC

		At least one dataset entry is now mandatory (to sum up the service content)	
0.32	6/2/2013	Corrections + Dataset_id now mandatory Separate particle_spectral_* parameters Dataproduct_type now encoded on 2-characters ID Reworked parameter/table attributes	SE, PLS, FH, BC
0.33	25/2/2013	Corrections Fixed case requirements	SE, PLS, FH, BC
0.34	11/6/2013	Index is back (from tests with the client) + Added ra/dec as optional parameters + fixed summary table (Appendix C)	SE, PLS, BC
0.35	10/11/2013	Finalized spectral conversions (App. A) Updated spatial coordinates definitions Added solar longitude as optional parameter	SE, PLS, BC
0.36	14/11/2013	Clarified spectral resolution limits Recovered Figures Added reference to IVOA thesaurus Added target_distance as optional parameter Added native_access_url/format as optional parameters + updated examples Added Appendix D, list of reserved keywords (ADQL functions)	SE, PLS, BC
0.37	29/01/2014	- Relaxed case constraint on target names (now uses standard spelling). - Completed appendix D with SQL reserved keywords - Fixed appendix C (parameter summary) TBD: Complete summary table (Appendix C) with possible values/sources ?	SE, PLS, BC
0.38a-e	2/01/2015	Reworked after publication in A&C, reviewers comments are reflected + discussions with M. Demleitner. - Added local_time as an option - element_name renamed feature_name for consistency - Ls renamed solar_longitude to remove ambiguities - index renamed granule_uid to remove ambiguities and avoir conflicts - reference renamed bib_reference, for conformity with SSA and ObsCore. - More details about spatial_frame_type: does not have to be constant in a service, must be included in any request that includes spatial coordinates - Some corrections to UCDs - Checked units	SE, PLS, BC
0.40	21/05/2015	Retropedaling from v0.38e so as to finalize v1 according to current data services and A&C publication [RD25]. Removed the following: - element_name renamed feature_name	SE

		for consistency - index renamed granule_uid to remove ambiguities and avoid conflicts - reference renamed bib_reference, for conformity with SSA and ObsCore. (solar_longitude maintained instead of Ls, which was not used in existing services) In addition, local_time now has *_min and *_max. TO DO: - reformat according to ObsCore doc - rewrite sections 4.4 and 5 (service answer) (& remove attributes?) Added latter (versions a/b/c/d/e): - Some clarifications in parameters definitions - Some corrections to UCDs in text and tables - Added Appendix D (q.rd example) - References to DaCHS' q.rd are maintained as examples only. - Access_format now introduces mime type, example values added	

Reference documents

[RD1] Planetary data access protocol (PDAP). IPDA draft 1.0 (16 April 2013)
https://planetarydata.org/projects/active-projects-2012-2013/PDAP%20Core%20Specification%20

[RD2] Keywords
RFC 2119: Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, ed. IETF (Internet Engineering Task Force), March 1997. Available at http://www.rfc-editor.org/rfc/rfc2119.txt

[RD3] Virtual Observatory Support Interface (VOSI)
http://ivoa.net/Documents/VOSI/20101206/index.html

[RD4] GAVO/ DaCHS implementation:
http://vo.ari.uni-heidelberg.de/docs/DaCHS/

[RD5] EPN data model version 1.18a (last version to date) can be found here:
http://www.europlanet-idis.fi/documents/public_documents/Data_Model_v1.18a.pdf

[RD6] TAP protocol
http://ivoa.net/Documents/TAP/

[RD7] ObsTAP and ObsCore
http://ivoa.net/Documents/ObsCore/

[RD8] UCD + UType concept
http://ivoa.net/Documents/cover/UCDlist-20070402.html

[RD10] IVOA Astronomical Data Query Language Version 2.00 http://ivoa.net/Documents/latest/ADQL.html

[RD11] Name resolver returning body official names and astronomical coordinates at a specific time:
http://vo.imcce.fr/webservices/ssodnet/?resolver

[RD12] Report of the IAU Working Group on Cartographic Coordinates and Rotational Elements: 2009
B. A. Archinal et al, Celest Mech Dyn Astr (2011) 109:101–135.
+ Erratum to: Reports of the IAU Working Group on Cartographic Coordinates and Rotational Elements: 2006 & 2009. B. A. Archinal et al, Celest Mech Dyn Astr (2011) 110:401–403.

[RD13] Space time and coordinate in IVOA
http://ivoa.net/Documents/latest/STC.html

[RD14] IVOA Registry interface
http://ivoa.net/Documents/RegistryInterface/

[RD15] Unit in the IVOA (current draft)
http://ivoa.net/Documents/VOUnits/

[RD16] EPN-RI Interoperable Data Access / Data Organization conventions, V0.0a1

[RD17] EPN-RI Coordinate systems, V0.2

[RD18] EPN-TAP documentation:
http://voparis-europlanet.obspm.fr/xml/TAPCore/

[RD19] The Committee on Small Body Nomenclature handles Minor Planet Names and Designations, Comet Names and Designations, Cross Listed Objects:
http://www.ss.astro.umd.edu/IAU/csbn/

[RD20] In addition, the IAU Working Group for Planetary System Nomenclature (WGPSN) defines feature names on planetary surfaces:
http://planetarynames.wr.usgs.gov/

[RD21] IAU Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites
http://astrogeology.usgs.gov/Page/groups/name/IAU-WGCCRE

[RD22] IAU nomenclature for object types:
http://planetarynames.wr.usgs.gov/Page/Planets

[RD23] EPN-TAP services: using TOPCAT as a client
http://voparis-europlanet.obspm.fr/utilities/Tuto_TopCat.pdf

[RD24] EPN client:
http://voparis-europlanet-new.obspm.fr/planetary/data/epn/query/all/

[RD25] EPN-TAP principle:
Erard S., P. Le Sidaner, B. Cecconi, J. Berthier, F. Henry, M. Molinaro, M. Giardino, N. Bourrel, N. André, M. Gangloff, C. Jacquey, F. Topf (2014) The EPN-TAP protocol for the Planetary Science Virtual Observatory. *Astronomy & Computing* 7-8, 52-61. http://arxiv.org/abs/1407.5738

[RD26] Europlanet VO, general presentation:
Erard S., P. Le Sidaner, B. Cecconi, J. Berthier, F. Henry, C. Chauvin, N. André, V. Génot, C. Jacquey, M. Gangloff, N. Bourrel, B. Schmitt, T. Capria, G. Chanteur (2014) Planetary Science Virtual Observatory architecture. *Astronomy & Computing* 7-8, 71-80. http://arxiv.org/abs/1407.4886

[RD27] SPASE Data Model:
http://www.spase-group.org/data/

Acronym list

ADQL	Astronomical Data Query Language
DaCHS	German VO (GAVO) Data Center Helper Suite: a framework to handle TAP data services
EPNCore	Set of core parameters from EPN-DM, mandatory for EPN-TAP compatibility
epn_core	Name of a view in a database which provides the EPN-TAP parameters to be queried. Required for EPN-TAP compatibility.
EPN-DM	Specific Data Model to describe Planetary Science data in Europlanet-VO
EPN-TAP	Specific protocol to access Planetary Science data in Europlanet-VO
Europlanet-RI	European Union funded program in FP7 (2009-2012)
FITS	Flexible Image Transport System: one of the basic data formats in Astronomy
IDIS	Integrated and Distributed Information Services: data handling activity in Europlanet-RI
IPDA	International Planetary Data Alliance
IVOA	International Virtual Observatory Alliance
ObsTAP	TAP protocol applied to the Observation Data Model of IVOA
ObsCore	Set of core parameters from the Observation Data Model of IVOA
PDAP	Planetary Data Access Protocol: protocol to access planetary data space archives, developed and maintained by IPDA
PDS	Planetary Data System: the de facto standard for Planetary Science space borne archives
PSR-DM	Planetary Science Resource Data Model: a more exhaustive DM for Planetary Science, completing EPN-DM
SAMP	Simple Application Messaging Protocol from IVOA
SPASE	Space Physics Archive Search and Extract
SSODnet	Solar System Open Database Network
TAP	Table Access Protocol: one of the protocols developed by the IVOA to access astronomical data
UCD	Unified Content Descriptor: define measured physical quantities in the IVOA
Utype	Description of data properties, in relation with a Data Model
VAMDC	Virtual Atomic and Molecular Data Center: a EU-funded program in FP7
VESPA	Virtual European Solar and Planetary Access The EPN-TAP/PDAP client and main user interface: http://vespa.obspm.fr
VOSI	Virtual Observatory Support Interface

1 - Introduction

EPN-TAP is a VO data access protocol designed to support Planetary Science data in the broadest sense. It is intended to access data services of various content, including space-borne, ground-based, experimental (laboratory), and simulated data. It is designed to describe data in many fields, from surface imaging to spectroscopy, atmospheric structure, electro-magnetic fields, and particle measurements. EPN-TAP is an essential part of the Planetary Science Virtual Observatory (VO), because no preexisting protocol was able to access such a large realm of data [RD26]. A description of principle has been published in [RD25], while the present document is intended to provide details for implementation.

The EPN-TAP protocol is directly derived from IVOA's Table Access Protocol (TAP) [RD6], a protocol to access data organized in tables, here adapted to Planetary Science. EPN-TAP is an extension of TAP with extra characterization derived from a Data Model — just like ObsTAP is an extension based on the Obscore Data Model [RD7].

The EPN Data Model is used to describe many types of Planetary Science data using a standard terminology [RD5]. EPN-TAP uses a subset of this terminology to define standard query parameters. This subset of the EPN Data Model is called EPNCore. Some parameters are adapted from ObsCore, but also from the PDAP protocol of IPDA [RD1] and from the SPASE protocol (Space Physics Archive Search and Extract) [RD27].

Since EPN-TAP is TAP compliant, the discovery of all EPN-TAP services can be performed using an IVOA registry. EPN-TAP services are described accurately by IVOA registries that include the TAPRegExt extension [RD26]. Once declared in a registry, EPN-TAP compliant data services are most efficiently queried with a specific EPN-TAP client such as the VESPA tool at VO-Paris [RD24].

This document describes the basis of EPN-TAP with a focus on data providers. EPN-TAP definition includes:
- A general framework to implement data services (SQL database, the presence of the epn_core view, etc).
- A set of parameters describing the resources and their content (the EPNCore DM), plus optional parameters and attributes.
- A convention to provide numeric parameters in standard form (units/scales, etc) for the query mechanism.
- A set of reference sources to encode the string parameters (target names, etc).
- A set of UCDs defining the parameters in use in the VO context.

Practical use cases are listed in Appendix B to support the reflection about EPN-TAP.

It is stressed that this version 1.0 of the EPN-TAP protocol was implemented as a demonstrator and on early EPN services, to assess the efficiency of this system. It is potentially deprecated and replaced by version 2.0 (currently under definition), which solves potential issues and inconsistencies related, e. g., to data description.

2 - Main concepts of EPN-TAP

EPN-TAP is an extension of IVOA TAP and is compliant with the TAP standard. It typically uses the TAP mechanism [6] with synchronous or asynchronous queries, and VOSI for capability and metadata access [3].

TAP is a protocol dedicated to access relational database tables. It uses ADQL (the Astronomical Data Query Language, [RD10]) to query the databases. To allow similar queries on all EPN-TAP services, we will assume that the EPNCore data model is implemented in the database as a view (i. e., as a table). In order to be accessed through EPN-TAP, all databases must therefore include a view called epn_core, which contains at least all the parameters described in section 4.2. The epn_core view mainly contains a list of the “granules” available in the database, typically an entry/line for each data element, and is used as a catalog of the accessible content. The parameters describing the granules are mostly related to data description and to the main axes of variation.

2.1 Axes description

In this system, the user writes his query on a client interface. The client sends a formatted query to the server. The server in turn looks for matches in the epn_core view and sends back an answer. This process is illustrated in Fig. 1. A standard situation is to search data located in space, time or spectral range, therefore to issue a query based on axis coordinates.

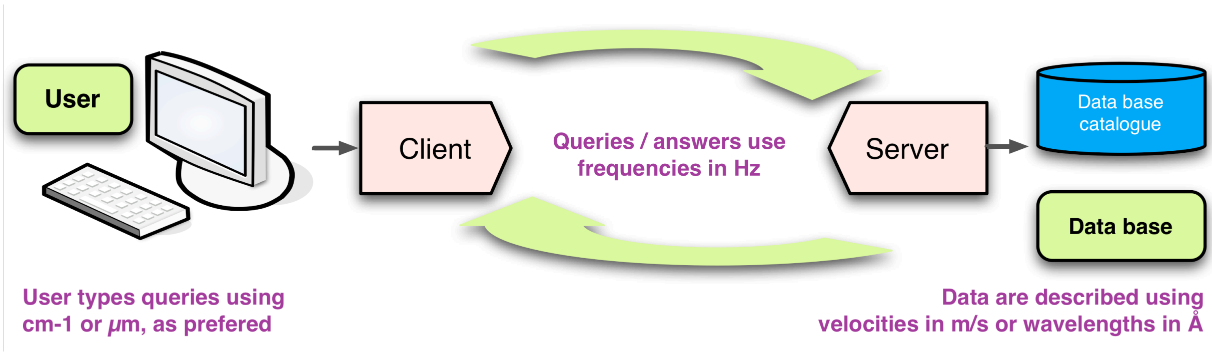

Fig. 1: Client/server general scheme for EPN-TAP

In order to handle the multiplicity of situations, several parameters are normalized in the protocol, regardless of the content of the databases. For instance, a spectroscopy database may provide measurements on a wavelength scale in microns, while the user wants to query the data on a wavenumber scale in cm^{-1} (Fig. 1). A common description should therefore be used, which should not interfere with the way the data are described, nor with the way the user wants to query the data.

The EPN-TAP standard defines the scales and units used for all parameters — e. g., the spectral axes are always described on a frequency scale in Hz. Since the databases do not necessarily use the standard scales/units internally, the epn_core view also has the function to provide the parameters in the expected units once for all. This avoids on-the-fly conversions on the database side, while the data themselves may remain in native form (Fig. 2). It is therefore essential that such transforms are exactly reciprocal on both sides of the query system (see Appendix A). A similar system is used for many parameters, e. g., time scales are provided in Julian Days.

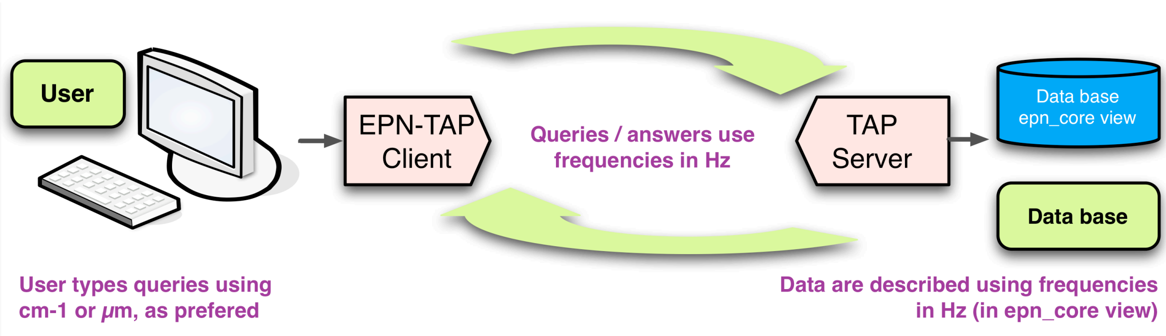

Fig. 2: Practical implementation for EPN-TAP services

The EPN-TAP protocol is closely related to the TAP protocol, and mainly differs by the definition of its core parameters. The server side relies on a general framework for TAP, while the client performs most EPN-specific operations and turns them into fully TAP-compliant queries, which can be handled directly by the database through ADQL. During the service set-up, the epn_core view of the database must therefore be declared so that all EPNCore parameters are presented according to the standard. In the previous example, if the spectral axis is provided in wavelengths in the database, the epn_core view must include a transformed version on a frequency scale in Hz. This view is used as an interface for the client, and is not normally seen by the users.

Parameter names are mostly used as tags to pass the values between the client and the server. Since they are used to handle a variety of situations, science fields, etc, they may not reflect the exact meaning of the parameters in the frame of a specific database. This again is not an issue, since parameter names are normally not seen by the users (depending on the client interface).

A particular situation arises with the spatial coordinates, because of the extreme diversity encountered in Planetary Science. In order to simply formulate a query, the general type of coordinate system (e.g. celestial coordinates, geographical coordinates, Cartesian coordinates in a volume…) must be known in advance. For this reason the description must be included in the column description of the TAP response [RD6] and in the metadata returned by the service.

Since TAP can only query the columns of a table, some important parameters have to be provided in columns even though they are constant throughout the service (e.g., spatial_frame_type).

2.2 Data structure

An EPN-TAP service can contain essentially four types of data: (a) scalar data fields in limited number; (b) data contained in one separated file (image, table...); (c) data spread on several separated files; (d) data computed on the fly. These situations may be handled as follows:

(a) The data may be included in the epn_core view as separated columns with specific, non-standard parameter names. In general dataproduct_type = ca (catalog) is appropriate, and no access_* parameter is needed in the epn_core view.
(b) A URL to the external file is provided on each line through the access_url parameter, so that the client can easily download the selected files. This description may be completed by the access_format, access_estsize, and preview_url parameters. The dataproduct_type parameter must be filled according to the data organization type (e.g., image, time_series...).
(c) A "main data product" must be identified, which is described as in (b). Additional data products are linked and described using parameter names derived consistently from the standard ones. "Preview_url" is actually a common example of such a situation. Other examples include images with associated ancillary data in separated files, referred to as e.g., ancillarydata_access_url, and alternative output format referred to as "native_access_url".
(d) The access_url must point to a computing system that will process the query, e.g. forwarding a query to a computing service with adapted parameters.

2.3 Data description

The epn_core view mainly contains a list of the "granules" available in the database, i. e., typically an entry/line for each data file. The structure, in particular the dataproduct_type, is not necessarily constant among all granules in the epn_core view.

In addition to granules, at least one "dataset" entry is required for each service. Parameters describing "datasets" provide the ranges encompassed by their elements/granules. "Datasets" and "granules" entries are identified using the resource_type parameter. A query on "dataset" may be used to return only global information on a service, without a long list of available data products, and is therefore the preferred access mode in discovery phase. For this reason, an EPN-TAP client will preferably default to resource_type = dataset. In the epn_core view, datasets are best located at the beginning for visibility (most VO clients only load a limited number of entries by default, so the last ones are often not visible).

Additional "datasets" can be defined inside the epn_core view. Such datasets consist in subsets of granules selected according to various criteria by the data provider. A complex PDS data set for example can be sliced into several subsets accessed independently through EPN-TAP. This allows data providers to make their data available in EPN-TAP without going through the burden of generating alternative versions of their databases. When several datasets are present, the dataset_id parameter will permit to restrain queries to identified datasets.

An important part of the service design is related to the identification of the granules, and is left to the data provider. The simplest situation corresponds to one entry per data file, but complex situations may call for other solutions. For instance, if an image contains both Mars and Phobos, the basic approach is to have one granule with the two target names stored in the target_name parameter. Alternatively, if the target is considered as the main entry, there could be two granules (Mars and Phobos) pointing to the same image file; this will permit to provide the coordinates relative to each body with no ambiguity (a similar situation may occur when the data files contain several data products of different types). A third possibility would be to combine the first two, and to define three granules pointing to the same image. Although there is no mandatory rule, this third possibility is in general not desirable: redundancy in the epn_core view will result in duplicate answers, which may be both confusing and unpractical for the user. Data providers will in general want to give answers as explicit as possible.

2.4 Service response

The server looks for lines of the epn_core view matching the query (Fig. 3). The answer is an excerpt of the table containing all the table columns, including the EPNCore parameters and the data, embedded in a VOTable. Data access is therefore provided according to the table definition.

Altogether, the epn_core view is composed of many fields (Fig. 3): all mandatory EPN-TAP parameters; possibly optional or extra parameters; data access information, either data embedded in the view or access information to data files. In addition, the service includes general metadata describing the table itself, which are included in the response.

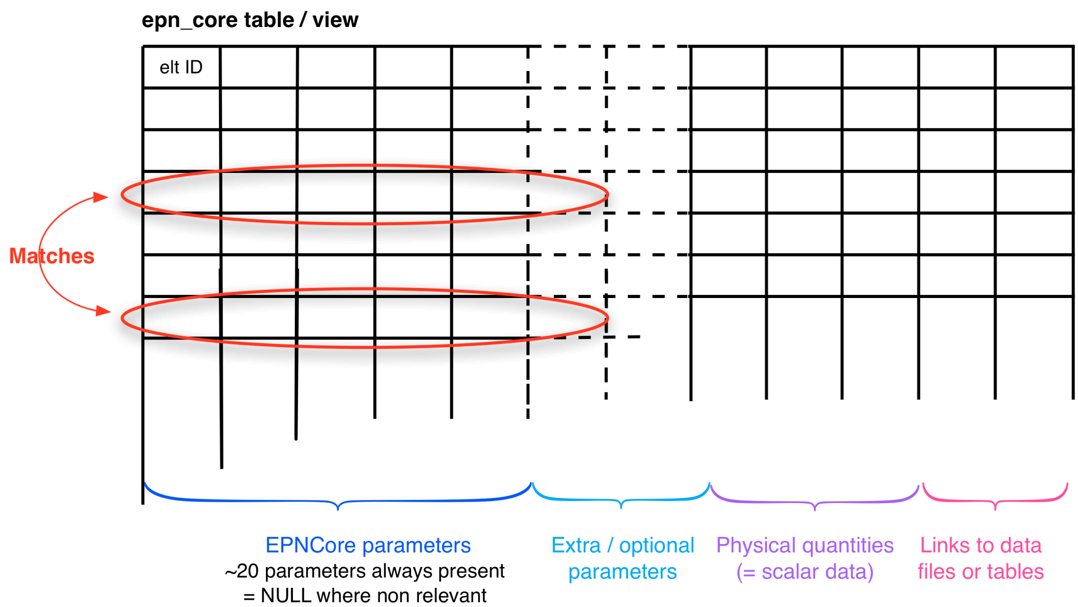

Fig. 3: Query of the epn_core view and return values

“Capabilities” of TAP services are accessible as described in VOSI. The capability query includes basic references to the service, including credits for the original data.

Any table field can be queried with TAP, including the data themselves when they are listed in the optional columns of an epn_core view. This mechanism provides a complete access to the data service (in contrast for example to the PDAP protocol v1).

2.5 Implementation

EPN-TAP services may be implemented in various ways. The first ones have been installed using the GAVO/DaCHS framework; possible alternatives include VO-Dance and Saada. Apart from the DaCHS installation document [RD4], tutorials to install EPN-TAP services using DaCHS are available on this page:

http://voparis-europlanet.obspm.fr/docum.shtml

In the DaCHS framework, services are defined in a file q.rd mapping the epn_core view (Appendix D), and in an xml file providing the service declaration to the registry.
The former contains a RESOURCE element, which essentially contains a TABLE element with ID ="epn_core". Both include META elements providing general information. Columns information is provided in COLUMN elements of the TABLE.
Templates containing generic definitions of the compulsory parameters are available to help defining new services.

2.6 Clients

Several ways to query EPN-TAP services are already available:
(a) The VO-Paris VESPA client can be used to query services declared in the OV-Paris registry, or to access local services not yet registered. The EPNCore parameters are entered in the user's preferred unit scales, and converted to EPN-TAP standard. Selected results can be sent to IVOA visualization tools through SAMP [RD24].
(b) The TopCat tool may be used as a low-level client to send general TAP queries to individual databases, visualize data, and make data available to other clients through SAMP [RD23].
(c) The DaCHS framework includes a client (ADQL query page) which permits to send general TAP queries to local databases individually.
(d) Taphandle can also send TAP queries (http://saada.u-strasbg.fr/taphandle/)
(e) Command line access is possible using tapsh (http://soft.g-vo.org/tapsh)

3 - Requirements for compliance

We will use the standard keywords of the W3C "MUST", "REQUIRED", "SHOULD", and "MAY" to describe the requirements [RD2]. A compliant TAP service has to verify all the MUST and REQUIRED statements. As far as possible SHOULD statements will be reached, and MAY statements can be considered as preferred options.

A "best practice" is also mentioned in several occasions as an option, e. g., concerning target names. In [RD2] formalism, the "best practice" is equivalent to SHOULD. It is separated here more clearly to stress that an EPN-TAP service can be queried when these practices are not honored, but that the benefit of VO interoperability would mostly be lost. The capacity to identify relevant data and to make the service interoperable relies on the use of standard conventions by the data providers. A service that does not use the standard IAU target names will not respond correctly to standard EPN queries, so its data would not be accessible to its potential users. "Best practices" are therefore strongly recommended in any case to take advantage of VO capacities. Practically, the only case when the "best practice" may not be honored is when setting up an EPN-TAP interface on a pre-existing database without updating its contents, and this is expected to be only a temporary situation.

EPN-TAP is an extension of IVOA TAP and is compliant with the TAP standard. It typically includes:

- TAP using ADQL [RD6].
- VOSI as capability and metadata [RD3]
- TAP queries can be synchronous or asynchronous. For specification of the asynchronous mode one can refer to the TAP document [RD6].

In addition, EPN-TAP also includes some constraints and restrictions:

- A table (or view) called epn_core must be present and must contain all the EPNCore mandatory parameters.
- All mandatory EPNCore parameters must be provided in the epn_core view using the standard units and scales, which may require some conversions (see Appendix A).
- All mandatory and optional string parameters in the epn_core view must be provided in lower cases — in practice, they must not be directly copied from the database but transferred using the function *lower(string)*. Exceptions are the "target_name" and "species" parameters, and parameters introducing URLs, filenames, which are case sensitive. Other, additional string parameters are case sensitive as well. The space character is not allowed in the values, but is used as a separator.
- There is one reserved value, which must be accepted for all parameters: NULL (in upper case, PostGreSQL standard). Parameters left empty in the epn_core view are handled as if they were filled will the NULL value.
- Some fields can contain multiple values, in particular when describing a dataset (e.g., target_name = Mars Phobos Deimos). Various values are separated by the space character.
- At least one "dataset" line must be present. It gives a summary of the service contents, with values encompassing those of all the granules.
- Some descriptive parameters provide information through strings (e.g. target_name, instrument_name...). The best practice is to use standard values/syntax/encoding. Reference lists for such parameters have been identified and are provided in the parameter descriptions below; these references are part of the EPN-TAP standard and should be used to ensure interoperability of the data services.
- Some parameters actually provide a description of other parameters, and should remain constant in the epn_core view (e. g. spatial_frame_type). Consistency checks are left to the data provider.

4 - EPN-TAP queries

A TAP query consists in looking for certain values of the parameters in the data table. Its arguments are therefore the parameters/columns of this table. Such queries act as a filter on the database contents, and return only the lines of the table matching the arguments.

An EPN-TAP client will typically send the query to all known services, and present a list of services providing a positive answer. It can also restrict to a set of selected services, e. g., those which provided an answer to a first query. Queries are therefore often run in two steps, starting with a discovery/exploration phase to identify services of possible interest (e.g., those containing spectroscopy of Mars), and then addressing a specific question (e.g., a particular spectral range and location at the surface). The overall list of EPN-TAP services used in the exploration phase is accessed from a registry.

The client must use the HTTP GET or POST protocols to send queries to services. The query is composed of the URL of the service, and ADQL language [RD10] is used to express the request. The TAP query is very generic and there is no mandatory parameter associated with it.

Example query:
*http://<server address>/tap/sync?REQUEST=doQuery&LANG=ADQL&QUERY=select * from epn_core where time_min > '2455197.5' and time_max < '2455927.5'*

Will return all kind of data from 2455197.5 (01/01/2010) to 2455927.5 (01/01/2012) in Julian days (target is not specified)

4.1 Service Behavior

EPNCore defines a set of parameters that must be handled by any EPN-TAP service even if they are not relevant for this service (see section 4.2 below). In this case, the corresponding fields in the epn_core view must be set to "NULL" or left empty. If the client sends a query using such a non-relevant parameter, the service must answer with no data (i.e., elements containing NULL for the query parameter will not be included in the answer, since they do not fit the query). Conversely, parameters omitted (or set to NULL) in the query will not be used to filter the data.

An EPN-TAP client may set a default value for some parameters, in particular for resource_type. A query using a single parameter resource_type = granule would reply with the complete list of granules / data files in the service, which is not optimal for resource exploration; the "dataset" value is more adapted in this case and may be the client's default.

Some parameters can be multivalued in the sense that the epn_core view can accommodate several values, in particular when related to datasets. The separator between values is always a space. To query such parameters, the "like" operator must always be used instead of the "=" operator. These fields include: target_name, target_class, instrument_name, instrument_host_name, measurement_type.

Example:
*http://<server address>/tap/sync/request=doquery & lang=adql & query=select * from epn_core where time_min between '2455197.5' and '2455927.5' and target_class like 'comet' and target_name like '1P'*

The service will return all data of any type for comet Halley (1P) from 2455197.5 (01/01/2010) to 2455927.5 (01/01/2012) in Julian days

Similarly, a single query can introduce multiple values for a given parameter. ADQL provides standard operations on parameters to combine possible conditions (and, or, like…) as well as parentheses. Standard ADQL wildcards are also implemented [RD10].

Example:
target_name like 'Mars' or target_name like 'Venus'

Return data on both Mars and Venus

Because of a limitation in ADQL language, most query parameters listed below are case insensitive, i.e. should be provided and queried in lower-case. In practice, applications and clients must present parameters using only lower-case characters, and strings in the epn_core view must be transferred in lower cases as described above. Case sensitive parameters are: target names, URLs, filenames, "species" and all non-standard parameters (i.e., defined for a particular service and not listed below).

EPN-TAP services may also contain parameters not included in EPNCore. All such parameters can be accessed using the corresponding names. Notice that names of ADQL functions are reserved and cannot be used as parameter names (see Appendix D). The information of the service table is accessible using VOSI [RD3]. All the metadata tables related to a data service can be obtained via:
HTTP GET http://<server address>/tap/tables

VOSI also provides information about general service capabilities (e.g., IVOA protocols supported by the service). The "capabilities" can be obtained via:
HTTP GET http://<server address>/tap/capabilities

Finally, "Availability" gives information on the current status of the service (up/down...):
HTTP GET http://<server address>/tap/availability
See VOSI document for details on the availability resource.

4.2 Compulsory parameters

Similarly to IVOA's ObsCore [RD7], EPNCore defines a set of parameters describing the EPN-TAP services. These parameters must all be present in the epn_core view of the database, and must be understood as they are defined below.

The parameters attributes must be included in the service response: numerical type, unit, UCD, and a short description string. With the DaCHS framework, this is provided in the "q.rd" file mapping the parameters (Appendix D).

All EPNCore parameters are listed and described in this section. Parameters are characterized using their UCDs and Utypes [RD8]. If compulsory parameters are not relevant or unknown, they must be present in the epn_core view with a value = NULL (or left empty).
Other parameters present in the data service may be queried by EPN-TAP through the same mechanism, but are not systematically supported.

Most numerical parameters are introduced with min and max values, to make interval comparisons possible. Whenever only one value is available (e. g., a unique acquisition time), it must be entered in both min/max fields.

Some of the compulsory parameters may correspond to different UCDs and Utypes depending on context (e. g., the first coordinate may be a distance or an angle). The UCDs and Utypes are part of the service description setup by the data provider. It is therefore the provider's responsibility to adjust the UCDs and Utypes of the service; caution in this matter will insure optimal match with external services. Multiple possibilities and solutions are described below. Whenever new values are required (e. g., to describe data file contents), it is a good practice to look for equivalent descriptions in [RD8] and [RD7].

4.2.1 Index

Name in epn_core view : index
Type : long
unit: unitless
Utype : ??
UCD : meta.id

This parameter provides a unique line number in the epn_core view, typically the entry number of the resource in the database.

This column is present to allow easy referencing of rows across multiple clients keeping the resource set (e.g., in SAMP). While this number should be stable as long as identical queries give identical results, there are no expectations of stability of this number when the total number of rows in the table changes (e.g., on a re-import of updated data).

4.2.2 Resource Type

Name in epn_core view : resource_type
Type : string
unit: unitless
Utype : Epn.ResourceType
UCD : meta.code.class

Defines the scope of the query. There are two possible values:
dataset and **granule**

The epn_core view lines correspond to individual data objects, and point to a single file (or group of files); this corresponds to the "granule" level of information, the smallest level described in the catalogue. A granule is therefore the smallest element reachable in a data service: a file, a group of associated files, a table entry, or some kind of data computed on the fly. This typically corresponds to a line in the epn_core view.

Subsets of granules may also be grouped together and summarized on a single line of the epn_core view, which defines a "dataset". The same EPN service may contain several datasets. Responses to queries related to granules or datasets are formally similar. The dataset field definitions are derived from the component granules in the most straightforward way (e.g., dataset time_min is the minimum time_min of included granules…).

In the epn_core view, at least one entry resource_type = dataset is required to provide a summary of the data available in the service. The access_url parameter may provide a URL access to a full dataset archive in one file. Data providers could omit this link, for instance in case of very large datasets they don't intend to distribute globally.

Other datasets may be present to describe granules grouped according to arbitrary criteria. Cross-reference between datasets and individual granules are provided through the dataset_id parameter.

An EPN-TAP client may default to resource_type = dataset. This will allow test queries to return information on datasets only, therefore limiting the number of answers. Although this parameter is not mandatory in queries, it makes little sense for the client to query both datasets and granules together.

4.2.3 Dataset ID

Name in epn_core view: dataset_id
Type : string
unit: dimensionless
Utype : ??
UCD : meta.id

The dataset_id parameter provides cross-references when the table includes several datasets (i.e., several lines with resource_type = dataset). It introduces a unique reference ID for these datasets; when applied to granules, it gives the reference to datasets that include the granule. IDs are composed of standard alphanumeric characters, with no space.
When only the mandatory dataset is present, this parameter can be set to 1 for all entries.

4.2.4 Data Product Type

Name in epn_core view: dataproduct_type
Type : string
unit: unitless
Utype : Epn.dataProductType
UCD : meta.code.class

The dataproduct_type parameter describes the high level scientific organization of the data product linked by the access_url parameter (see below), or directly included in the table (in which case the value is most likely 'catalog'). EPNCore currently defines several types listed below. The data provider should select the type most adapted to his data. In complex situations (e. g., when a file contains several data products), several types can be used to describe the same granule — although using several granules to describe the file content may be a better solution.
In the epn_core view, these types are identified by a 2-characters ID, so that multivalued queries are unambiguous. Possible IDs listed below with their meaning:

* im = image: scalar field with two spatial axes, or association of several such fields, e.g., images with multiple color planes, from multichannel or filter cameras. Maps of planetary

surfaces are considered as images.

* sp = spectrum: measurements organized primarily along a spectral axis, e.g., a series of radiance spectra.

* ds = dynamic_spectrum: consecutive spectral measurements through time, organized as a time series.

* sc = spectral_cube: sets of spectral measurements with 1 or 2D spatial coverage, e.g., imaging spectroscopy. The choice between image and spectral_cube is dictated by the characteristics of the instrument (which dimension is most resolved).

* pr = profile: scalar or vectorial measurements along 1 spatial dimension, e.g., atmospheric profiles, atmospheric paths, sub-surface profiles...

* vo = volume: other measurements with 3 spatial dimensions, e.g., internal or atmospheric structures.

* mo = movie: sets of chronological 2D spatial measurements

* cu = cube: multidimensional data with 3 or more axes, e.g., all that is not described by other 3D data types such as spectral cube or volume. This is mostly intended to accommodate unusual data with multiple dimensions.

* ts = time_series: measurements organized primarily as a function of time (with exception of dynamical spectra and movies, i.e. usually a scalar quantity). Space-borne dust detector measurements are a typical example of a time series.

* ca = catalog: can be a list of events, a catalog of object parameters, a list of features.... e. g., a list of asteroid properties. It can be limited to scalar quantities, and possibly limited to a single element (i.e., one catalog entry). Time_series, Profile, and Catalog are essentially tables of scalar values. In Time_series the primary key is time; in Profile it is altitude or distance; in Catalog, it may be a qualitative parameter (name, ID...).

* sv = spatial_vector: list of summit coordinates defining a vector, e.g., vector information from a GIS, spatial footprints...

Usage:
*select * from epn_core where resource_type = 'granule' and dataproduct_type like 'im'*

will return only image data

4.2.5 Target Name

Name in epn_core view: target_name
Type : string
unit: unitless
Utype : Epn.TargetName
UCD : meta.id;src

The target_name element identifies a target by name or ID. The target may be any Solar System body, exoplanet, planetary sample, or meteorite, plus in some cases astronomical objects. Any other feature (craters, regions, atmospheric layers…) must be named using the optional element_name parameter (see 4.3.3).

The best practice is to use the official name of the target as defined by IAU [RD19]. This parameter is case sensitive (mixing lower/upper cases) and all values must use the standard spelling and case. Data providers must understand that services that do not use the IAU names might not be accessible by the clients. Conversely, users should be aware that some services containing data of interest might not be visible, if they do not use the recommended IAU nomenclature for planetary bodies. The SSODnet name resolver provided by VOParis-IMCCE may help data providers (and users as well) to handle multiple denominations [RD11].

Other best practices are listed below:
The Exoplanet Encyclopedia provides a complete list of currently known extrasolar planets:
http://exoplanet.eu/

Meteorite catalogs can be found here:
http://www.nhm.ac.uk/research-curation/research/projects/metcat/search/indexsing.dsml
http://www.lpi.usra.edu/meteor/index.php

The catalog of lunar samples is available here:
http://www.lpi.usra.edu/lunar/samples/

Other planetary samples are listed in topical web sites, e.g. samples from the Stardust mission are described here:
http://curator.jsc.nasa.gov/stardust/catalog/

Usage:
*select * from epn_core where target_name like 'Ceres' or target_name like 'Vesta' and target _type like 'dwarf_planet' or target_class like 'asteroid'*

Will return data only from 1 Ceres or 4 Vesta (see ADQL syntax). Complex queries may also include parentheses

Example
1P is the official IAU name for comet Halley.

4.2.6 Target Class

Name in epn_core view: target_class
Type : enum string
unit: unitless
Utype : Epn.TargetClass
UCD : meta.code.class;src

The target_class element identifies the type of a named target. A target is defined without ambiguity by a couple of parameters: target_class and target_name (although some targets may have no proper name).

EPNCore defines the possible values for target_class:
asteroid, dwarf_planet, planet, satellite
(types from IAU list [RD22])
comet, exoplanet, interplanetary_medium, ring, sample, sky, spacecraft, spacejunk, star
(extra types defined for EPN)

Usage:
Any target has a unique target type.
“interplanetary_medium” refers in particular to interplanetary dust.
“sample” refers to lunar or planetary samples, to meteorites, but also to terrestrial samples, e.g., in laboratory studies.
“satellite” stands for natural satellites only - other cases are handled though spacecraft or spacejunk.
“star” is used typically for calibration targets, and for the Sun.
“sky” may be used for other celestial bodies, usually referred to by their sky coordinates. It also includes the Interstellar Medium.

4.2.7 Time Range

Name in epn_core view: time_min, time_max
Type : double
unit: 'd' — Julian day
time_min:
Utype : Char.TimeAxis.Coverage.Bounds.Limits.Interval.StartTime
possibly: Utype : Epn.Time.time_min
UCD : time.start
time_max:
Utype : Char.TimeAxis.Coverage.Bounds.Limits.Interval.StopTime
possibly: Utype : Epn.Time.time_max
UCD : time.end

NB2: pour les Utypes, on pointe vers le DM EPNcore non ?
C’est dans EPNcore, qui faudrait repointer vers Chararacterization ou autre, Pierre ?

The time parameters provide the date and time of acquisition.
In EPN-TAP, the time parameters are always provided in UTC and formatted in Julian days (expressed as a double precision float). Although ObsCore uses Modified JD, EPNCore uses JD to avoid ambiguity with time origin. With double precision floats, the accuracy is on the order of 1 ms, which is considered sufficient to identify data of interest (the initial accuracy is preserved in the data itself).
The two values min/max permit to handle long periods. Whenever acquisition time is a scalar (rather than an interval), both time_min and time_max must contain the same value in the table. There is no limiting value to this parameter.

Examples:
*http://<server address>/tap/sync/request=doquery & lang=adql & query=select * from*

epn_core where time_min > '2455197.5' and time_max < '2455927.5'

Will search data described by a time range

*http://<server address>/tap/sync/request=doquery & lang=adql & query=select * from epn_core where time_min between '2455197.5' and '2455927.5'*

Will search data described by a start time parameter

Although time is provided in UTC, it is not necessarily measured on ground; e.g. spacecraft on-board times are acceptable if they are provided on a UTC scale. Therefore times may need to be corrected for light path in order to be compared with other events / datasets. The location where time is measured is provided though the Time_Origin parameter, which is defined at service level (see section 5).
Service providers may want to use non-compulsory parameters to accommodate additive, specially formatted time scales such as native on-board time (see section 5). The information of the time_min/time_max parameters is however greatly recommended, as it is the only simple way to relate independent datasets.

4.2.8 Time Sampling Step

Name in epn_core view : time_sampling_step_min, time_sampling_step_max
Type : double
unit: 's'
time_sampling_step_min
Utype : Epn.Time.Time_sampling_step_min
UCD : time.interval;stat.min
time_sampling_step_max
Utype : Epn.Time.Time_sampling_step_max
UCD : time.interval;stat.max

This parameter provides the sampling step for measurements of dynamical phenomena, and for computations. This is the time between 2 successive measurements or data, which is mostly relevant when the measurements are regularly spaced. This may also be used as a query parameter, e.g., for ephemeris computations.
This parameter is intended to allow the user to search for time-resolved observations of dynamic phenomena.

4.2.9 Exposure Time

Name in epn_core view: time_exp_min, time_exp_max
Type : double
unit: 's'
time_exp_min
Utype : Epn.Time.Time_exp_min
UCD : time.duration;stat.min
time_exp_max
Utype : Epn.Time.Time_exp_min
UCD : time.duration;stat.max

This parameter corresponds to the integration time of measurements. This time is usually shorter than the time_sampling_step if both are present.

4.2.10 Spectral Range

Name in epn_core view: spectral_range_min, spectral_range_max
Type : double
unit: 'Hz'
spectral_range_min
Utype : Epn.Spectral.Spectral_range_min
UCD: em.freq;stat.min
spectral_range_max
Utype : Epn.Spectral.Spectral_range_max
UCD: em.freq;stat.max

The spectral_range parameters define the upper and lower bounds of the spectral domain of the data. As mentioned previously, this quantity is expressed on a frequency scale in Hertz. Conversions to the native unit are provided in Appendix A.
The spectral range and associated parameters only apply to electromagnetic waves. See the optional parameters particle_spectral_* for particle energy or mass detection.

4.2.11 Spectral Sampling Step

Name in epn_core view: spectral_sampling_step_min, spectral_sampling_step_max
Type : double
unit: 'Hz'
spectral_sampling_step_min
Utype: Epn.Spectral.Spectral_sampling_step_min
UCD: em.freq.step;stat.min
spectral_sampling_step_max
Utype: Epn.Spectral.Spectral_sampling_step_max
UCD: em.freq.step;stat.max
em.freq.step ça n'existe pas... et ce n'est pas vraiment comme ça qu'il faudrait le mettre à mon avis. Je vais y réflechir

The spectral_sampling_step is the spectral separation between the centers of two adjacent filters or channels. Like all spectral_* quantities, it is expressed on a frequency scale in Hz. Conversions to the native unit are provided in Appendix A. The min and max values are expected to correspond to both ends of the spectral range if there is any ambiguity.
This parameter is mostly intended to provide an order of magnitude, e.g., to distinguish between grating spectrometers and Fourier spectrometers, or between observations related to surfaces or atmospheres. In can also help distinguishing between Nyqvist and sub-Nyqvist sampling rates.

4.2.12 Spectral Resolution

Name in epn_core view : spectral_resolution_min, spectral_resolution_max
Type : double
unit: 'Hz'

spectral_resolution_min
Utype : Epn.Spectral.Spectral_resolution_min
UCD : spect.resolution;stat.min
spectral_resolution_max
Utype : Epn.Spectral.Spectral_resolution_max
UCD : spect.resolution;stat.max

(check that spect.resolution is actually OK)

The spectral_resolution parameters correspond to the spectral bandwidth used for the measurement (Full Width at Half Maximum). In case of a filter camera this is the filter bandwidth; in case of a spectrometer this is the spectral resolution per se. The min and max values are expected to correspond to both ends of the spectral range if there is any ambiguity.
This parameter is mostly intended to provide an order of magnitude, e.g. to distinguish between grating spectrometers and filter cameras.

4.2.13 Spatial Coordinates (c1, c2, c3)

Name in epn_core view : c1min, c2min, c3min, c1max, c2max, c3max
Type : vector of double
unit: depending on context (deg, m...)

Utype : char.SpatialAxis.Coverage
c1min, c2min, c3min
Utype:
Epn.Spatial.Spatial_range.c1min
Epn.Spatial.Spatial_range.c2min
Epn.Spatial.Spatial_range.c3min
UCD: pos;stat.min ou obs.field;stat.min (?) (Ca peut être les deux...)
c1max, c2max, c3max
Utype:
Epn.Spatial.Spatial_range.c1max
Epn.Spatial.Spatial_range.c2max
Epn.Spatial.Spatial_range.c3max
UCD: pos;stat.max ou obs.field;stat.max (?)

This parameter provides up to three spatial coordinates of the measured target. The coordinates depend on the spatial frame type defined below. All services should handle three spatial coordinates, even if the third one is always set to NULL. Note that the c3 parameter is related to the observed area; the target distance (e. g., geocentric distance for ground based observations, or spacecraft distance) is best introduced by the optional parameter “target_distance”.

The query will be made on the coordinate system proposed by the provider, and no conversion is expected (see spatial_frame_type below). More precise description of the coordinate system is given in the response metadata. Descriptions for EPN-TAP are provided in [RD17].

Secondary coordinates can be introduced using additional axes, e.g., c1 and c2 providing central longitude and latitude of a planetary disk, and extra RA / DEC columns providing location on the sky at this moment.

4.2.14 Spatial Resolution

Name in epn_core view:
c1_resol_min, c2_resol_min, c3_resol_min, c1_resol_max, c2_resol_max, c3_resol_max
Type: double
unit: depending on context (deg, m, ...) — same as spatial_range
c1_resol_min, c2_resol_min, c3_resol_min
Utype:
Epn.Spatial.Spatial_resolution.c1_resol_min
Epn.Spatial.Spatial_resolution.c2_resol_min
Epn.Spatial.Spatial_resolution.c3_resol_min
UCD: pos.resolution;stat.min
c1_resol_max, c2_resol_max, c3_resol_max
Utype:
Epn.Spatial.Spatial_resolution.c1_resol_max
Epn.Spatial.Spatial_resolution.c2_resol_max
Epn.Spatial.Spatial_resolution.c3_resol_max
UCD: pos.resolution;stat.max

This parameter provides a simple estimate of resolution, either the FWHM of the PFS on the sky (in degrees), or the pixel size on a surface (in m), depending on spatial_frame_type.

The client frontend may propose more appropriate units to the user, depending on context (e.g., angular resolution in mas, distance in m...).

4.2.15 Spatial Frame Type

Name in epn_core view: spatial_frame_type
Type : string
unit: unitless
Utype : Epn.Spatial.Spatial_frame_type
UCD : meta.code.class;pos.frame

Provides the "flavor" of the coordinate system, which defines the nature of the spatial coordinates (c1,c2,c3). The possible types are described below:

celestial: 2D angles on the sky, e. g., right ascension c1 and declination c2 + possibly distance from origin c3 – although this is a special case of spherical frame, the order is different.

body: 2D angles on a rotating body: longitude c1 and latitude c2 + possibly a z c3 coordinate. The best practice is to follow the IAU 2009 planetocentric convention [RD12], in particular eastward longitudes and a north pole located on the north side of the invariant plane of the Solar System for planets and satellites (see [RD12] for small bodies, and Annex A for details). The Z coordinate is by default the distance counted from the center of mass.
The spatial_coordinate_description and spatial_origin attributes allow the data provider to indicate different conventions, e. g., to indicate a planetographic frame, or to use altitude above a reference surface as the third coordinate. It is stressed however that using other frames will make comparisons between datasets more difficult.

cartesian: (x,y,z) as (c1,c2,c3). This includes spatial coordinates given in pixels.
cylindrical: (r, theta, z) as (c1,c2,c3); angles are defined in degrees.
spherical: (r, theta, phi) as (c1,c2,c3); angles are defined as in usual spherical systems (E longitude, zenith angle/colatitude), in degrees. If the data are related to the sky, “celestial” coordinates with RA/Dec must be used.
healpix: (H, K) as (c1,c2).

This parameter, although related to the specific coordinate system in use, is mostly intended to identify the nature of the coordinates handled by the service (e. g., angles versus distances). This parameter is provided as a column of the epn_core view, to ensure it can be queried through the basic TAP mechanism. However, it is essentially an attribute of the coordinates and is in general expected to remain constant along the table. This parameter should be provided in any query that includes spatial coordinates. Whenever additional coordinates are provided, they must be stored in extra columns of the table. If several different frames are mixed to provide the main coordinates, the use of different datasets may help clarify the situation; datasets entries must sum up all spatial frame types included in the dataset. At any rate, easy access to the data must be considered during the design of the service.
Ranges and specific definitions vary with the actual frame in use, and are discussed in Appendix A.

4.2.16 Incidence Angle

Name in epn_core view: incidence_min, incidence_max
Type : double
unit: 'deg'
incidence_min
Utype : Epn.View_angle.Incidence_angle_min
UCD pos.posAng;stat.min
incidence_max
Utype : Epn.View_angle.Incidence_angle_max
UCD pos.posAng;stat.max

The incidence angle parameters define the upper and lower bounds of the incidence angle variation in the data (also known as Solar Zenith Angle). This is always indicated in decimal degrees, and may range from -180 to 180° (with 0° indicating the normal to the surface). Incidence and emergence angles may be counted relative to the normal of the ellipsoid model, or to the local normal (e. g., using a 3D shape model). In case the two systems are included in the data, these keywords introduce the values relative to the ellipsoid (local values may be available through non-compulsory parameters).

4.2.17 Emergence Angle

Name in epn_core view: emergence_min, emergence_max
Type : double
unit: 'deg'
emergence_min
Utype : Epn.View_angle.Emergence_angle_min
UCD pos.posAng;stat.min
emergence_max
Utype : Epn.View_angle.Emergence_angle_max

UCD pos.posAng;stat.max

The emergence angle parameters define the upper and lower bounds of the emergence angle variation in the data (viewing angle). This is always indicated in decimal degrees, and may range from -180 to 180° (with 0° indicating the normal to the surface).
Incidence and emergence angles may be counted relative to the normal of the ellipsoid model, or to the local normal (e. g., using a 3D shape model). In case the two systems are included in the data, these keywords introduce the values relative to the ellipsoid (local values may be available through non-compulsory parameters).

4.2.18 Phase Angle

Name in epn_core view: phase_min, phase_max
Type : double
unit: 'deg'
phase_min
Utype : Epn.View_angle.Phase_angle_min
UCD pos.phaseAng;stat.min
phase_max
Utype : Epn.View_angle.Phase_angle_max
UCD pos.phaseAng;stat.max

The phase angle parameters define the upper and lower bounds of the phase angle variation in the data (scattering angle - 180°, or angle light source-target-observer). This is always indicated in decimal degrees, and may range from -180 to 180° (with 0° corresponding to opposition, i. e., light source in the back of the observer). Negative values may refer, e. g., to geometry before opposition, depending on context.
Phase, incidence and emergence are partly related:

$$\mathrm{abs}(i - e) < \varphi < i + e$$

If the azimuth angle α is provided instead of the phase angle, the latter can be derived from knowledge of the three angles:

$$\cos \varphi = \cos i \cos e + \cos \alpha \sin i \sin e$$

4.2.19 Instrument Host Name

Name in epn_core view: instrument_host_name
Type : string
unit: unitless
Utype : Provenance.ObsConfig.Facility.name
Utype : Epn.Instrument_host_name ???
UCD : meta.id;instr.obsty

This parameter provides the name of the observatory or spacecraft that performed the measurements. The best practice is to use names from the lists indicated below. A list of host names should be provided for integrated data sets.

For ground-based observations, the reference is the list of IAU observatory codes:
http://www.minorplanetcenter.net/iau/lists/ObsCodesF.html

However, this list is not intended to include all ground-based observatories, and a complement still needs to be identified (including e. g. radio-telescopes).
A reasonably complete list of radio-telescopes is available here:
http://en.wikipedia.org/wiki/List_of_radio_telescopes

Other open Q:
- are IAU ID eligible?
- granularity is very heterogeneous (e.g. one single entry for Paranal or Mauna Kea, but many for siding spring).
- Some entry related to obs programs (New Horizon KBO search from various sites)

Concerning space-borne data, the most complete list of international planetary missions and orbital observatories is found here (included in a complete list of space missions with ID):
http://nssdc.gsfc.nasa.gov/nmc/
Planetary missions are also listed here:
http://nssdc.gsfc.nasa.gov/planetary/chronology.html

Alternatively, the PDS dictionary defines values for many mission names:
http://pds.nasa.gov/tools/dictionary.shtml
Other mission names are supported by the SPICE system, but only as ID codes:
http://www-int.stsci.edu/~sontag/spicedocs/req/naif_ids.html
(TBC – Spice is unambiguous but only uses IDs, PDS values are explicit but somewhat arbitrary)
In the epn_core view, the acronym is preferred to the full name to avoid long strings and related errors. Both values can be provided (e.g., HST + Hubble Space Telescope).

4.2.20 Instrument Name

Name in epn_core view: instrument_name
Type : string
unit: unitless
UTYPE : Provenance.ObsConfig.Instrument.name
Utype : Epn.Instrument_name ???
UCD meta.id;instr

Identifies the instrument(s) that acquired the data. A list of instruments should be provided for integrated datasets.

Service providers are invited to include multiple values for instrument name, e.g., complete name + usual acronym. This will allow queries on either "VISIBLE AND INFRARED THERMAL IMAGING SPECTROMETER" or VIRTIS to produce the same reply.

Concerning space-borne data, the most complete list of international planetary missions and orbital observatories is found here:
http://nssdc.gsfc.nasa.gov/nmc/
Instruments on board planetary missions in particular are listed here:
http://nssdc.gsfc.nasa.gov/nmc/experimentSearch.do

4.2.21 Measurement Type

Name in epn_core view: measurement_type
Type : string
unit: unitless
Utype : Char.ObservableAxis.ucd
Utype : Epn.Measurement_type
UCD : meta.ucd

The measurement_type parameter defines the physical quantities contained in the data, using UCDs. It relates to the reported quantity, not to the type of experiment. Therefore only UCD related to physical quantities can be used; e.g., phys.absorption;em.opt.I is eligible, while stellar_occultation is not.

The provider should use the “UCD1+” list from IVOA as a reference, and should extend it only when necessary [RD8]:

http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.pdf

The measurement_type parameter is used to search data relevant to a certain field. Whenever several quantities are comprised in the granule, the measurement_type parameter must therefore refer to all these quantities, including multiple UCDs if needed. Multiple values are separated by spaces.

Datasets entries must sum up all measurement types included in the dataset. This is a handy way to identify the science content of a complete service.

Extra UCDs will be proposed/requested to IVOA. In the meantime, an extended list of UCDs will be made available in the EPN-DM document (in progress).

Examples:
For images in general, the relevant UCD is obs.image, whatever the calibration level.
Alt : phot.flux would also be OK?
For spectra phot.flux.density describes a radiance vector - while the spectral vector is described by UCDs em.wvl, em.freq or em.energy, and the related error is described by stat.error;phot.flux.density.

4.3 Optional parameters

EPN-TAP can query parameters not included in the EPNCore. Some of these parameters are defined precisely but are relevant only to very specific data services. Those are not mandatory, but they must be implemented as defined in this section when present. Beside, the names of optional parameters are reserved for this particular usage and must not be used to introduce other quantities.
Whenever constant throughout the service, some of these parameters can be defined at the table level, instead of the granule level (i. e., in the description of the epn_core view rather than as an column).

4.3.1 Access Reference (access_url)

Utype : Obs.Access.Reference
Utype: EpnResponse.File_info.AccessURL
UCD : meta.ref.url;meta.file

The data of interest are often stored in a file, not in the table itself. In this very usual case, the access_url parameter provides a complete path to the data products on the network, so that they are accessible for download by plotting or processing tools. All URLs in the epn_core view are case sensitive and must provide an actual link. However, the link may be the output of a script on the server, in which case this parameter provides a call to the script with adequate arguments (e.g. Titan atmospheric profiles service). In any case, this parameter must link to the actual data, not to a file of metadata nor to a document.
Whenever the data consists in a few scalar fields, this parameter may be replaced by parameters providing the data itself (e.g. mass, in a table providing the masses of Solar System bodies).

When the data is spread among several files, variations on this parameter may be used. The data provider must identify a “main data product” which is linked through this parameter (and the related ones below). This will allow the plotting tools to download some data in any case. E. g., a service may provide links to calibrated images, plus raw data and ancillary information for every granule; the main product will probably be the calibrated image, but other files can be described using non-standard parameters such as ancillarydata_access_url – beware that such fields may not be easily accessible by the client or tools.

4.3.2 Access Format (access_format)

Utype : Obs.Access.Format
Utype: EpnResponse.File_info.Access_format
UCD : meta.code.mime

Access_format provides the format of the data file linked through the access_url parameter. This parameter is intended for automated processing of results by tools, not as a search parameter (the dataproduct_type parameter must be used for this purpose, e.g., when looking for images).
The data may be stored in their native format, and no format conversion is required to set up an EPN-TAP service. This field can therefore include reference to unusual formats, although those may not be handled by plotting tools but only in a specific environment. Consistently with ObsCore, possible values are MIME-types written in lower cases and are listed in Appendix F.

4.3.3 Estimated Size (access_estsize)

Utype : Obs.Access.Size
Type : integer
unit: kbyte
Utype : EpnResponse.File_info.Access_estsize
UCD : phys.size;meta.file (TBC with CDS)

The access_estsize field provides an order of magnitude (in kilobytes) of the file available via the corresponding URL. It is intended to provide an indication that can help to tune download functionalities in an application, depending on data volume and transfer bit rate.

4.3.4 Preview Reference (preview_url)

Name in epn_core view: preview_url
Type : string – free format
unit: unitless
Utype : Obs.Access.Reference
Utype : EpnResponse.??? Not in DM?
UCD : meta.ref.url;meta.file

The preview_url parameter contains the URL of a reduced version of the data product used for quick-look purpose (e.g. a small Jpeg image). This may be handy in the case of big data files or unusual data formats, to facilitate data selection by the user. Besides, the EPN-TAP client uses this preview for on-line quick-look, which therefore provides important added value to a service. The preferred formats include Jpeg and PNG (which should be handled easily by a basic viewer). If several previews are provided, variations on this name can be used. All URLs in the epn_core view are case sensitive and must provide an actual link.

4.3.5 Native access Reference (native_access_url)

Utype : Obs.Access.Reference
Utype: EpnResponse.File_info.AccessURL
UCD : meta.ref.url;meta.file

The native_access_url parameter provides access to the original version of the data file, in addition to the formatted version provided by access_url. This is typically used by a service that reformats the data files on the fly. For instance, the Titan atmospheric profile service normally provides VOtable written by the server, but the original ascii files can still be retrieved though this keyword. If a script is used to provide the access_url link, the native_access_url link may be provided by the same script using a different output format.
This parameter may also refer to original PDS3 files when the data have been converted to VOTables.

4.3.6 Native access format (native_access_format)

Utype : Obs.Access.Format
Utype: EpnResponse.File_info.Access_format
UCD : meta.code.mime

The native_access_format parameter provides description of the files accessed through the native_access_url parameter.

4.3.7 File Name (file_name)

Name in epn_core view: file_name
Type : string – free format
unit: unitless
Utype : ?
UCD : meta.id;meta.file

The file_name parameter introduces the name of the *data* file, with no path information. In many data services, the file name encodes the most relevant metadata and may be a very handy access mechanism at least for specialists. All filenames in the epn_core view are case sensitive and must reflect an actual filename.

4.3.8 Species

Name in epn_core view: species
Type : string – standard formulas only
unit: unitless
Utype : ?
Utype : Not in DM?
UCD : meta.id;phys.atmol

The species parameter introduces the chemical species of interest in simple data services. The formatting is very basic and simply uses the standard formula in ascii, e. g., H2O for water, CO2 for carbon dioxide or Fe for iron. This is the only query parameter that is provided in case sensitive form, using the standard chemical notation. This format can only accommodate atoms and simple molecular species, and does not support isotopic variations.
An example application is related to atmospheric composition: a table providing the vertical abundances of many gaseous species with altitude. All columns are abundances and are described by the same measurement_type parameter. Only the use of the “species” parameter (together with the column name itself) allows identifying the various species and accessing the requested information.
If the data contain one column per species, it is recommended to also include the species in the column name (e.g., H2O_abundance).
If more elaborated compositional information must be included, the use of another parameter providing InChiKeys is recommended.

4.3.9 Element Name

Name in epn_core view: element_name
Type : string – free format
unit: unitless
Utype : Not in DM?
UCD : meta.id;pos

The element_name parameter introduces a supplementary name to provide more details about the observed target. It is intended in particular to accommodate a local name (crater, surface feature, region name...) whereas target_name is reserved to describe the whole body (mars, moon, ceres...). The best practice is to use the official features name defined by IAU [RD20] when relevant.
The target_region parameter (see below) also provides additional information on the target, but is aimed at indicating the global scope of a database (e.g., atmospheric layer, internal structure...).

4.3.10 Reference

Name in epn_core view: reference

Type : string
unit: unitless
Utype : Not in DM?
UCD : meta.bib

The reference parameter introduces an individual bibliographic reference at granule level. This may be required, e. g., if the resource is a compilation of data from various origins.
This is best provided as a Bibcode as used e. g. in ADS.

4.3.11 Celestial coordinates (RA/Dec)

Name in epn_core view: ra, dec
Type : float
unit: ? See Cone Search
Utype : ?
UCD ?

If fixed sky coordinates of the target are provided in the view in addition to standard coordinates, they must be stored in parameters named ra and dec. This may document the location of a planet in a celestial image, while the main coordinates c1/c2 are used to describe the observed area.

4.3.12 Solar longitude (Ls)

Name in epn_core view: Ls
Type: float
unit: 'deg'
Utype: ?
UCD: pos.posAngle

Solar longitude (a.k.a. heliocentric longitude, or ecliptic longitude of the Sun, traditionally noted Ls) is the Sun-Planet vector angle counted from the planet position at N hemisphere spring equinox. It provides a measurement of season.
Ls = 90° corresponds to the northern summer solstice, Ls = 180° to the northern autumn equinox, and Ls = 90° to the northern winter solstice. Although it is most usually applied to Mars and Titan (using Saturn's Ls), this notion can be enlarged to any planetary body without ambiguity.
This should not be confused with the true anomaly of the body, which is the same angle counted from the perihelion position.

4.3.13 Local time

Name in epn_core view: local_time_min, local_time_max
Type : float
unit: 'hours'
local_time_min:
Utype : ?
UCD: time.phase;stat.min
local_time_max:
Utype : ?
UCD: time.phase;stat.max

Planetary observations may be documented using the local time at the surface, i. e. the location of the solar meridian normalized to 24. This parameter is provided in unit of target rotation divided by 24 and is measured from local midnight (ranges from 0 to 24, should increase with time at a given location). It is provided in decimal hours.

4.3.14 Target Distance

Name in epn_core view: target_distance
Type: double
unit: dimensionless
Utype: ?
UCD: pos.distance

The target_distance parameter introduces the distance of the observer to the observed area, not to be confused with a vertical dimension provided by c3. For ground-based observations, this is the geocentric distance of the target. For space borne data, this is the spacecraft-target distance.

4.3.15 Particle Spectral Type

Name in epn_core view: particle_spectral_type
Type: string
unit: dimensionless
Utype: ?
UCD ?

This parameter and the following ones are related to the spectral distribution of particles only (see the spectral_* parameters for electro-magnetic waves).
The particle_spectral_type parameter introduces the type of axis in use: either energy (provided in eV), mass (in amu), or mass/charge ratio (in amu/qe).

4.3.16 Particle Spectral Range

Name in epn_core view: particle_spectral_range_min, particle_spectral_range_max
Type : double
unit: 'eV', 'amu', or 'amu/qe'
particle_spectral_range_min
Utype : ?
UCD : phys.energy;phys.part;stat.min
particle_spectral_range_max
Utype : ?
UCD : particle phys.energy;phys.part;stat.max

The particle_spectral_range parameters define the upper and lower bounds of the spectral domain for particles. Depending on the particle_spectral_type parameter, this quantity is expressed on an energy, mass, or mass/charge scale, with respective units eV, amu, or amu/qe. Conversions to the native unit are provided in Appendix A.

4.3.17 Particle Spectral Sampling Step

Name in epn_core view: particle_spectral_sampling_step_min, particle_spectral_sampling_step_max
Type : double
unit: 'eV', 'amu', or 'amu/qe'
particle_spectral_sampling_step_min
Utype: ?
UCD: ?
particle_spectral_sampling_step_max
Utype: ?
UCD: ?

The particle_spectral_sampling_step parameters provide the spectral separation between measurements, in the same scale and unit as particle_spectral_range.
Conversions to the native unit are provided in Appendix A. This parameter is mostly intended to provide an order of magnitude.

4.3.18 Particle Spectral Resolution

Name in epn_core view : particle_spectral_resolution_min, particle_spectral_resolution_max
Type : double
unit: 'eV', 'amu', or 'amu/qe'
particle_spectral_resolution_min
Utype : ?
UCD : spect.resolution;stat.min
particle_spectral_resolution_max
Utype : ?
UCD : spect.resolution;stat.max

The particle_spectral_resolution parameters correspond to the actual resolution of the measurements, and are provided in the same scale and unit as particle_spectral_range.
This parameter is mostly intended to provide an order of magnitude.

4.4 Parameter attributes

The columns of the epn_core view can be described more precisely using either optional parameters (in which case the value can change from granule to granule), or columns attributes (which value remains constant along the table).

Only the parameters of the epn_core view can be used for data selection in a query, therefore important attributes must be stored as parameters even when they remain constant throughout the table (which is often the case, e. g., for spatial_frame_type). A convenient way to get the broad properties of a service is to send a query limited to datasets, with no further parameter.

The client can also grab information from other sources. However, such attributes can only be included in the service response VOtable to document the output. VOSI supports only some attributes for each column: numerical type, unit, UCD, and description (free field); in DACHS, these are declared in the file “q.rd” defining the service. The registry can also include some properties or attributes of the overall service.

We discussed the possibility to add a second table epn_desc containing parameter attributes in the q.rd file. Apparently DaCHS could handle this with no major problem, TBC (for version 2.0…)

4.4.1 Processing level

EPN expression: processing_level
Type : integer
Utype PSR:processingLevel
Utype : EpnResponse.Complementary_return_info.Processing_level
UCD : meta.code;obs.calib

In the framework of EPN-TAP, this parameter is intended to provide the user with a quick evaluation of data "usability". Several classifications are in use in different contexts, as summarized in the table below. EPN-TAP uses the CODMAC levels (IDs coded as integers). "Partially calibrated" datasets are in general considered as not calibrated, but this evaluation is up to the data provider depending on context. "Ancillary" data include all extra information documenting the measurements, e. g., coordinates or geometry files. Although it may be more consistent to separate calibration levels in different data services, several levels can be included in the same service (in particular calibrated and ancillary data). Only one value can be accommodated in this field, so the most advanced level (1-5) should be used when several levels are available.

(Compilation of information from PSA & ObsCore documents)

CODMAC level / EPN-TAP	PSA level	NASA level	PRODUCT_TYPE (PDS3/PSA)	ObsTAP	Description (from PSA, with comments)
1 (raw)	1a		UDR	Level 0	Unprocessed Data Record (low-level encoding, e.g. telemetry from a spacecraft instrument. Normally available only to the original team)
2 (edited)	1b	0	EDR	Level 1 (std data format)	Experiment Data Record (often referred to as "raw data": decommutated, but still affected by instrumental effects)
3 (calibrated)	2	1A	RDR	Level2	Reduced Data Record ("calibrated" in physical units)
4 (resampled)		1B	REFDR		Reformatted Data Record (mosaics or composite of several observing sessions, involving some level of data fusion)
5 (derived)	3	2-5	DDR	Level3	Derived Data Record (result of data analysis, directly usable by other communities with no further processing)
6 (ancillary)			ANCDR		Ancillary Data Record (extra data specifically supporting a data set, such as coordinates, geometry…)

This quantity may be provided as a column attribute when the data are imbedded in the epn_core view, or linked as external URL (it then applies to access_url and derived parameters).

4.4.2 Data unit and dimension

UCD: meta.unit ?

Although the epn_core parameters are provided in standard scales/units, service output values will be provided in native form, and associated with physical units that need to be identified — in particular to send the data to VO plotting tools.
Native units are provided in the file defining the service parameters (e.g., "q.rd" file under DaCHS), and are reported as field descriptors in the output VOtable.
??? Beware that this is not related to a parameter in general, but to a quantity available in a file, perhaps mixed with other ones
We also need a sort of measurement_type (e.g., to send data to VOspec).

This field is case sensitive (mixing lower/upper cases) and follows the International System of Units (SI) standard. For ease of notation, the caret "^" indicating powers of ten, and the multiplicative dot "." are both optional.
— are we sure of this ? This seems an unnecessary complication for the client.

Example:
"Jansky" or "W.m^-2.Hz^-1" or "Wm-2Hz-1"
are equivalent

4.4.3 Description of coordinate frame

Spatial_coordinate_description
Spatial_origin

These two parameters provide description of the spatial frame(s) in use, as introduced by the spatial_frame_type parameter. This is mostly relevant when this parameter is constant throughout the service.
Possible values are detailed in [RD17], which is partly adapted from STC [RD13]

Examples (TBC)
BODY, Mars_IAU2000
ICRS, Geocenter

This appears in the "description" element of the spatial_frame_type parameter in the q.rd file ??? —TBC ; the name of the specific frame (first parameter) should appear there, but what of the second one? Then: what if spatial_frame_type is not constant?

4.4.4 Time Origin

This attribute states where the time is measured, and is expected to remain constant throughout the service. This knowledge is required to cross-correlate event-based observations, in particular to indicate light-path differences. It is expected to remain constant along the table, and applies to the time_min and time_max parameters.

Possible values from time_origin are:
Earth, (solar system bodies name), (spacecraft name)

This may appear in the “description” element of the time_range parameter, e.g. in the q.rd file for a DaCHS server. (in practice, used as an optional parameter)

4.5 Service properties

Some properties are defined at the level of the entire service. A part of this information (in particular the one related to publisher and credits) is available only in the registry.

Most of these fields must be included in the service response however, and therefore an EPN-TAP client must be able to grab this information from the registry.

List of general properties of the service or table (all these TBC):
+ Can we use names already used as parameter/column names?

4.5.1 Service_protocol

Introduces a constant string = “EPN-TAP”. The associated value provides the protocol version number.
This is available in the registry — However it is also included in the current service response??

4.5.2 Title / Name

Provides the service/table title.
This is available in the registry — However it is also included in the current service response??

4.5.3 Creation_date

Provide the date when the service was last updated.
This is available in the registry — However it is also included in the current service response??

4.5.4 Access_url (global)

Provides network access to the EPN-TAP service.
This is available in the registry — However it is also included in the current service response??

4.5.5 ReferenceURL (global)

Provides bibliographic references for the entire dataset or service.
This is available in the registry — However it is also included in the current service response??
This is best provided as a Bibcode as used e. g. in ADS.

4.5.6 Curation

Provides credits for the data service. This elements includes three fields:
- “creator” introduces the responsible/coordinator of the science content (PI).
- “contributors” introduces the contributors to the contents, including at technical level.
- “publisher” introduces the ID of the publishing entity.

Example:

```
<curation>
   <publisher ivo-id="ivo://vopdc.obspm/lesia">VO-Paris Data Centre - LESIA</publisher>
   <creator>
      <name>L. Lamy (Observatoire de Paris - LESIA)</name>
   </creator>
  <contributor>F. Henry, VO-Paris Data Centre</contributor>
   <date>2012-06-06</date>
<version>1</version>
<contact>
  <name>L. Lamy</name>
  <address>Observatoire de Paris, 5 place Jules Jansen, 92195 Meudon, France</address>
  <email>laurent.lamy@obspm.fr</email>
  <telephone>+33145077661</telephone>
</contact>
</curation>
```

4.5.7 Target_region

Type : string
unit: unitless
ucd : meta.main => src.class ?

This parameter optionally identifies the region of interest for the resource, in complement to target_name. This parameter only introduces generic regions, not specific local names, which must be handled using the element_name parameter (see examples above).
The best practice is to take the values from standard sources:

- IVOA thesaurus: http://www.ivoa.net/rdf/Vocabularies/vocabularies-20091007/IVOAT/
- IAU thesaurus http://www.mso.anu.edu.au/library/thesaurus/
  + another version: http://www.vocabularyserver.com/trex/en/

The latter seems more recent and more complete (although the interface is not practical)
- Spase dictionary http://www.spase-group.org/

Example:
"atmosphere", "surface", "ionosphere"

The same sources are used for the declaration file in the registry.

4.5.8 Data_access_info

Provides indications relative to data handling. May introduce mime type, data structure, data reader… (TBC). It is not mandatory.

Should also tell:
- File format and how to read it
  => access_format and readerURL in the DM would do
- where to find physical quantities in the file (data of interest + axes) – this is not handled in VO plotting tools => the Virtis/Aladin demonstrator is a use case for this.
- some parameters required to indicate the file area to be read (e.g. which data product in a PDS file, which extension/column in a FITS…)
- What is/are the main axis?

Hum… is this only related to workflows later in the process?

4.6 Application to special services

EPN-TAP, because it is directly derived from TAP, may not be optimal to query computational services. Simulated data are accessible the same way as observational data, but the simulation parameters may be difficult to access through this mechanism. Similarly, experimental data are accessible but experimental setup and sample descriptions may be hard to reach (VAMDC-TAP may be a better choice to handle this kind of data).

Access to various data structures can be assessed through the Use Cases listed in Appendix B.

Tabular data are by construction more easily handled with EPN-TAP.
If the dataset mostly consists in files, EPN-TAP tells nothing about the file structure. As a consequence, the user cannot plot the files automatically if it is not a standard (e.g., image) format.

Time Scale

Name : time_scale
Type : string
unit: unitless
Utype : stc.timeScaleType
ucd : time.scale

This parameter defines which time scale is to be used in the answer, for instance when querying an ephemeris server. The way this parameter is handled is left to the service provider, but the actual value must be included in the output VOTable.
This parameter does not affect the time range included in the query, which must always be expressed using UTC – this is more a query option than a parameter.

Other time scales defined in STC [RD13] can be used:
- TT : Terrestrial Time: the basis for ephemeris
- TDB : Barycentric Dynamic Time: the independent variable in planetary ephemeris; time at the Solar System barycenter, synchronous with TT on an annual basis; sometimes called TEB
- TCG : Terrestrial Coordinate Time

- TCB : Barycentric Coordinate Time; runs slower than TDB but is consistent with physical constants
- TAI : International Atomic Time; runs 32.184 s behind TT
- UTC : Coordinated Universal Time; currently (2006) runs 33 leap seconds behind TAI
- GPS : Global Positioning System's time scale; runs 19 s behind TAI, 51.184 s behind TT

5 - Service response

The response of the service is formatted as a VOTable, which must comply with the VOTable standard, version 1.2 or higher. Additional outputs may be provided by specific frameworks.

5.1 - Service response metadata

The VOTable must contain a RESOURCE element with the attribute *type="results"* containing a single TABLE element with the results of the query. Additional RESOURCE elements may be present, but the usage of any such element is not defined here and the TAP client may not use them.

The RESOURCE element must contain several INFO elements:
• An INFO with the attribute *name="QUERY_STATUS"* and a *value* attribute which must contain one of the following:
"OK": the query executed successfully, and a result table is included in the resource. This does not imply that data are actually retrieved (i.e., no data may fulfill the query).
"ERROR": an error was detected at the level of the TAP protocol, or the query failed to execute. The content of the INFO element may provide the error description.

• Another INFO element with the attribute *name="SERVICE_PROTOCOL"* contains the identification of the protocol (EPN-TAP) and returns the version number supported by the service through its *value* attribute.

The content of the INFO elements should be a message suitable for display to the user. See [RD6] for more details.

Example:

```
<?xml version="1.0"?>
<VOTABLE version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.2
http://www.ivoa.net/xml/VOTable/VOTable-1.2.xsd">
  <RESOURCE type="results">
   <INFO name="QUERY_STATUS" value="OK"> </INFO>
   <INFO name="SERVICE_PROTOCOL" value="1.0">EPN-TAP</INFO>
   <TABLE>
     ...
   </TABLE>
 </RESOURCE>
</VOTABLE>
```

If no result fulfills the query, the TABLE element must be present and empty (i. e., the TABLE element has no DATA element).
The VOTable must also include basic information concerning the origin of the data, in particular an acknowledgement sentence to be included in possible publications (this is stored and read in the registry). Clients may also include an acknowledgement sentence relative to the EPN-TAP system.

• Currently (dec 2012), the VOTable retrieved from the client includes other INFO elements containing:
- the name of the data server

```
<INFO name="server" value="http://voparis-tap.obspm.fr"/>
```

- the EPN-TAP query

```
<INFO name="query" value="SELECT * FROM vo_mars.epn_core WHERE 1 = 1 AND
target_name LIKE '%%Mars%%' AND resource_type = 'granule'"/>
```

- 2 descriptions of the service and the table— TBC!!!

```
<INFO name="src_res" value="Contains traces from resource vo_mars/q">
Ozone profile in mars atmosphere data from Alain Sarkissian</INFO>

<INFO name="src_table" value="Contains traces from table
vo_mars.epn_core">ozone in mars atmosphere.</INFO>
```

- A credits note: (this is currently named "copyright", should be "credits")

```
<INFO name="copyright" value="Content note">Blabla... </INFO>
```

+ Add date/time of query

5.2 - Query response metadata

Some metadata are returned together with the data for user's information. They include the parameters described in this section.

In the current built of DaCHS [RD4] some of this information may only be stored in the registry itself, and cannot be included in the output VOTable. It has to be grabbed by the client separately, and included in the output available on the client's response page.

5.2.1 Service information / table metadata

Information about the origin of data, property, acknowledgements... should come with the data themselves to trace and notify their origin.
To minimize the length of the response, only the publisher, the reference of publication (bibcode), the title, and the IVO dataset identifier will be returned. No parameter is compulsory in the query response.

The IVOA unique identifier will allow accessing all information from the registry (See IVOA

registry interface [RD14]): publisher, owner, origin, acknowledgements....
Service providers should declare their services in the registry system used by EPN, similar to the IVOA one — details will be provided in a future document.

5.2.1.a Publisher

Type : string
UCD : meta.ref

A short string identifying the publishing institute of the data (e.g., a data archive or data center) or an indexing service such as ADS.

5.2.1.b Reference

Type : string
UCD meta.bib

Provides a bibliographic reference, which can be an URL or a Bibcode.

5.2.1.c Service_title / service_name ?

Type : string
UCD : meta.title

This parameter provides the title/name of the data service.

Also include any ID of the resource?

5.2.2 Data description fields

Some data-related parameters must also be provided as part of the general description of the service. These must be readily available to the client in order to formulate correctly even basic queries. — TBC: this information has to be made available to the *client*.

5.2.2.a Spatial_frame_type

Name : spatial_frame_type
Type : string
UCD : pos.frame

The type of spatial frame used for the main coordinates (c1,c2,c3).

5.2.2b Time Scale

Name : time_scale
Type : string
Utype : stc.timeScaleType
ucd : time.scale

Specifies the time scale used in the response. Although this is expected to be UTC for data services, it may be different for instance when querying an ephemeris server. It should then correspond to the time_scale option provided in the query.

5.2.2c Measurement Type

Name : measurement_type
Type : string
Utype : Char.ObservableAxis.ucd

UCD : meta.ucd

A summary of all measurement types present in the service.

Any other one ?? ...

5.3 - Response data

The data area of the output VOTable contains the subset of rows from epn_core view that matches the query. The data can therefore be either linked with an access URL or directly embedded in the response VOtable if short enough. The fields corresponding to these parameters may have associated UCD, unit and a description.

If no result fulfills the query, the TABLE element must be present and empty (i.e., the TABLE element has no DATA element). Otherwise, it may be encoded in binary using base64 scheme (see VOTable properties).

The issue of format conversion is not addressed here. Concerning PDS3 data (which may be difficult to deal with), it may be convenient to read the data on the server and transfer them either in a VOTable or in a temporary FITS or ascii file. Such functionalities are server properties and are left to the data provider. – No, this is on the client side in fact

Output format can be any VOTable standard? I.e., xml table in ascii, binary encoded, or FITS file?

(SE, to be studied latter):
- Answer should include an indication of data structure (i.e. where to find values in a file...)

Appendix A: Unit conversions

The epn_core view must provide all quantities in the EPN-TAP conventional scales and units to make universal queries possible across different datasets. This does not involve any conversion in the database itself, though.

On the other side, an EPN-TAP client should ideally allow the user to enter his preferred scales and units, and convert them to the EPN-TAP standard using the symmetrical conversions.

The EPN-TAP client uses exact conversions as described below, using double precision constants. To maintain consistency and accuracy when accessing data provided in units different from the conventional ones, data providers should use the exact reciprocal conversions when preparing the epn_core view for their services.

A.1 Spectral axes

The EPN-TAP convention is to provide spectral quantities as frequencies measured in Hz.
The conversion is performed assuming propagation in vacuum, and the applicable conversion coefficients are provided in the table below. It is stressed that during conversions the min/max values may be permuted. In the epn_core view, min/max values always refer to the frequency scale.

Applicable equations:

$\lambda = c / f$

$u = 10^{-2} / \lambda = 10^{-2} * f / c$ (spectroscopy convention)

$E = hc / \lambda = h * f$

where:

f = frequency in Hertz ($Hz = s^{-1}$)

λ = wavelength in meters (m) [more usually given in µm or nm]

u = wavenumber in cm^{-1}

E = energy in electronVolts (eV)

Two constants are required for conversions. The values given below must be used, and the results declared as double precision floats for consistency.

- The speed of light; the exact value from Bureau des Poids et Mesures is

$c = 2.99792458 \times 10^{8}$ m/s

$= 1\,079\,252\,850$ km/h

- Plank's constant; the NIST/CODATA 2010 recommended values are

$h = 6.626\,069\,57 \times 10^{-34}$ J·s

$= 4.135667517 \times 10^{-15}$ eV·s

$hc = 1.98644568 \times 10^{-25}$ J·m

Approx. conversion factors (use exact ones)	Wavelength λ (nm)	Wavenumber u (cm^{-1})	Energy E (eV)
To frequency f (Hz)	$3 \times 10^{17} / \lambda$	3×10^{10} u	2.42×10^{14} E
From frequency f (Hz)	$3 \times 10^{17} / f$	3.33×10^{-11} f	4.1×10^{-15} f

Conversion from velocity to frequency

In some fields, spectral positions in the vicinity of a spectral line are given in terms of equivalent radial velocity (in km s^{-1}). The observed frequency *f* and the radial velocity *v* are related using two different approximations, depending on the domain.

The optical convention is:
$f = f_0 \cdot 1 / [1 + v / c]$
where c denotes the speed of light, and f_0 is the rest frequency of the observed spectral line.
The radio convention is often used at longer wavelengths:
$f_{rad} = f_0 [1 - v_{rad} / c]$

Spectral positions provided as velocities should be converted in frequencies (in Hz) in the epn_core view using the most adapted relation.
(the EPN-TAP client is not expected to provide conversion from velocity, since this would involve the input of a reference frequency).

A.2 Spatial axes

Although there is some flexibility, the main coordinates c1/c2/c3 are expected to document the area observed on the target. In particular c3 normally provides an altitude (or depth) over a surface (if c1/c2 are longitudes and latitudes), or the target distance on celestial images (if c1/c2 are right ascensions and declinations). Deciding what best describes the data is left to the data provider – but the purpose of these parameters is of course to allow easy identification of specific data products among many datasets.

Native axes

Data are projected in a frame related to the instrument or acquisition process.
Typically: X/Y for a camera, X/time for an imaging spectrometer
Should correspond to (TBC):
Spatial_frame_type = Cartesian
Spatial_coordinate_description = native
This type of description is reserved to uncalibrated data sets or experimental measurements.

Spatial frame type = celestial

In the epn_core view:
RA / DEC
TBD
deg or h & deg?

Whenever distances in the solar System are provided, the standard conversion coefficient from astronomical units is:
1 au = 149597870.7 km (IAU 2012 definition value)

Spatial frame type = body
In the epn_core view North latitudes are positive, and the coordinate system is always right-handed:
- Longitudes are provided in East-handed convention (longitudes ranging from 0 to 360 degrees eastward).
- Latitudes follow IAU conventions:
North pole is located in the northern celestial hemisphere for planets and big satellites; all small bodies are defined to have direct rotation (i.e., North pole is in the southern celestial hemisphere if rotation is retrograde).

C1min/max provide westernmost/easternmost longitudes of the observed area to avoid ambiguities when crossing the prime meridian.
C2min/max introduce southernmost/northernmost latitudes.

The transform from W to E convention reads like this (IDL syntax):

```
longE= -longW
ind=where(longE lt 0.)
if (ind(0) ge 0) then longE(ind)=longE(ind)+360
```

An EPN-TAP client should typically provide the possibility to enter either E or W longitudes, and convert them to the East-handed EPN-TAP convention.
If the data provider wants to include westward longitudes in the epn_core view, those must be introduced in an additional column.

The exact coordinate system is identified through the Spatial_coordinate_description and Spatial_origin parameters.

TBC for other systems

A.3 Time axes

In the epn_core view:
Times are provided in seconds.
Dates are provided as Julian days — many conversion tools are available. Julian days must be defined as double precision floats to maintain accuracy.

The place where time is measured is provided through the Time_origin parameter. Depending on this place (Earth, target, spacecraft…), a light-path correction may be required. PDS datasets implicitly use time measured on-board the spacecraft.

A client should typically accept either Julian days or calendar dates and convert them to Julian days. To preserve the possibility to include historical datasets, the conversion should apply at least from the XIX century.

If the time is not provided in a standard scale and cannot be converted easily into JD, the data service cannot use the main time axis. This concerns in particular non-calibrated data sets where time is provided in arbitrary scale, e.g. as a spacecraft elapsed time (SCET). In such cases, the best solution is to use a secondary time axis stored in a service-specific parameter. This axis will not be available for cross searches in general.

Appendix B: Use cases

This appendix contains a list of actual use cases to support the development of EPN-TAP. They are not necessarily implemented.

B.1 Tabular

• A list of asteroid properties
Example: size and mass of large asteroids
This is a table with one line / entry. Each line contains all the metadata and data relevant to an object. Each line is for a different target.
Parameters include target name/class, no localization or time.
Data consist in several scalar quantities included in the table (several columns)
The granule is one line, the dataset is the table itself (+ references).

• A list of Martian craters
Example: List of lobate ejecta craters on Mars by F. Costard.
A table providing few data for each object, one line / entry but there is only one target as defined above (Mars).
Parameters are: longitude/latitude, no time.
Data are: ejecta extension, type, ID or crater name (several columns)
The granule is one line, the dataset is the table itself (+ references).

• Ulysses/URAP Thermal Noise Time Series (as proposed by CDPP)
(Example 9.3 of EPN-DM doc [RD5]).
Scalar measurements through time, averaged.
Data are 6 scalar quantities + three “support parameters” (ie reconstructed data): Sun distance + heliographic coordinates. Those are actually EPNcore query parameters.
Hence, the data set is a simple table providing 6 scalar quantities for each time step.

B.2 Several related tables / files

• Vertical atmospheric profiles
Example: Titan database at VOParis
A list of vertical profiles of P / T / mixing ratio of 10 main species as a function of altitude.
Metadata are location of profiles (longitude/latitude), time… + perhaps a season parameter (= ls) stored as an extra column in the list
The granules are separated tables (or files) containing either composition of temperature profiles in individual locations.
The access_url refers to a php script that writes a VOtable on the fly. The native_access_url parameter may be used in this case to provide a direct link to the original ascii file (with header), or a call to the same script with a different output format option. The native output is intended to share the data files with other researchers out of the realm of the VO. The client

still provides search functions, but the data can be retrieved in native form for inclusion in a non-VO processing environment.

• STEREO/waves Level 2 data (Example 9.1 of EPN-DM doc [RD5]) seems to fall here.
It would do if all daily files are indexed in a general catalogue. If they are concatenated, it would belong to B.1.

• Cassini/RPWS/HFR/SKR Dynamic Spectra
(Example 9.2 of EPN-DM doc [RD5]) also seem to fall in this category
The list is an ordered time table
Parameter is time only (?)
Data include 8 scalar values and 4 spectra for each time step
Dataset metadata include frequency table, etc.
Granules are described as time slots extracted from the index table... It seems that a more consistent approach would be to define the granule as a set of data for a given time step.

Such cases do not appear very different from a series of files (section B.3 below).

B.3 Files

• Simple imaging database
Example: BDIP at LESIA
A list of images defined by target, observation time... + specific parameters (phase angle, target size, orientation...)
Parameters are target, observation time, location on the sky, observatory...
Data are the images + the specific scalar parameters included in the list
The granules are the images, which may be available in several formats (ie: several versions of the images may be available).
Several coordinate systems are used: RA-DEC for planet position in the sky, planetary coordinates for sub-terrestrial and sub-solar points, and visible regions. The coordinates of interest are clearly those of the sub-terrestrial point, on which the requests will normally be performed, so the spatial_frame_type is “body”. RA-DEC coordinates are provided as optional parameters.

• Simple spectral database
Example: IKS at LESIA/SBN PDS
A list of spectra of a single target through time/distance from the target: comet Halley as seen from the Vega spacecraft during approach.
Parameters include spectral range / resolution (constant, except for a single file providing a synthetic spectrum in the long wavelengths channel), distance of observation...
The original ascii files are converted to VOTables (wavelength/radiance vectors + header documenting the observation).
The spacecraft-target distance is provided as an optional parameter (it does not fit in the c3 parameter because it does not describe the observed region). Because the observed region of the surface is unknown, the whole coverage is provided in the coordinate parameters.

• Laboratory spectroscopy database
Example: MROCRISM spectral library (PDS3)

A list of spectra of mineral defined by sample / mineralogical class / spectral range / origin
Parameters include spectral range / resolution (almost constant)… and do not provide adequate description of the data.
Data include the spectral files + detailed description of measurement and samples.
The granules are formatted files + possibly separated labels containing complete information of data.
If we can't use target / target class for a minimal description of the samples, it is impossible to query the data on relevant parameters! Even so, most parameters of interest are not accessible through EPN-TAP (i.e., particle size).
This is therefore a good example of why we need to be able to query non-EPN_core parameters!
There is also an interesting issue with data format: the files are PDS3 ascii tables, which can't be accessed easily by IVOA plotting tools; they also include a variable number of columns (2 to 4, depending if accuracy in X and Y is included).

• Observational db including support data
Example: Virtis/Venus-Express archive (PDS3)
A list of observing sessions defined by time / location (longitude/latitude) / instrumental setup and observing conditions. Several instrumental channels are included. Several calibration levels are included.
Parameters are time / location / target / spectral range / integration time…
Data are raw + calibrated + geometry files + preview images + possibly derived products. All files have complex content (measurements + wavelength table, uncertainty…) and may vary in structure (internally described in PDS or FITS header).
Data also include "support parameters" which do not belong to the EPNCore: instrument channel, functioning mode…
The granules are calibrated data file + geometry file + preview (+ possibly raw/derived data file)
The EPNcore parameters may be sufficient to query the database efficiently, but the type of information retrieved must be described.

An alternative is to allow the provider to propose several data services from his dataset, therefore using several epn_core views (and different service declarations): one addressing the M-channel, the other one the H-channel; one for raw data, the other one for calibrated data… At any rate, calibrated data and geometry files must be handled together.

B.4 Computational

• Ephemeris server
Example: Miriade service
A list of parameters computed on the fly as time series.
Virtual data set (only metadata are defined).
Parameters are time / time scale / target / time step…
The granules are tables of computed values

Appendix C: epn_core parameter summary

This appendix contains a list of parameters extracted from the first actual EPN-TAP services. In the current GAVO/DaCHS implementation, these parameters are declared in a table definition file called the q_<service>.rd. Compulsory parameters must be declared with the present descriptions. Additional, specific, parameters may be included in this file with their corresponding description. Quantities describing the service itself can also be used as optional parameters if they apply to individual granules. Standard values of the parameters are listed in this document. A basic template is available on line.

Name	**Class**	**Unit**	**Description**	**UCD**
EPNCore mandatory parameters				
index	Long		Internal table row index	meta.id
resource_type	String		Can be dataset or granule	meta.code.class
dataset_id	String		Dataset identification & granule reference	meta.id
dataproduct_type	String		Organization of the data product, from enumerated list	meta.code.class
target_name	String		Standard name of target (from a list depending on target type), case sensitive	meta.id;src
target_class	String		Type of target, from enumerated list	meta.code.class;src
time_min	Float /double	d	Acquisition start time (in JD)	time.start
time_max	Float /double	d	Acquisition stop time (in JD)	time.end
time_sampling_step _min	Float	s	Min time sampling step	time.interval;stat.min
time_sampling_step _max	Float	s	Max time sampling step	time.interval;stat.max
time_exp_min	Float	s	Min integration time	time.duration;stat.min
time_exp_max	Float	s	Max integration time	time.duration;stat.max
spectral_range_min	Float	Hz	Min spectral range (frequency)	em.freq;stat.min
spectral_range_max	Float	Hz	Max spectral range (frequency)	em.freq;stat.max
spectral_sampling_step_min	Float	Hz	min spectral sampling step	em.freq.step;stat.min (not in list)
spectral_sampling_s	Float	Hz	Max spectral	em.freq.step;stat.max

tep_max			sampling step	(not in list)
spectral_resolution_min	Float	Hz	Min spectral resolution	spect.resolution;stat.min
spectral_resolution_max	Float	Hz	Max spectral resolution	spect.resolution;stat.max
c1min	Float	(1)	Min of first coordinate	Pos;stat.min
c1max	Float	(1)	Max of first coordinate	Pos;stat.max
c2min	Float	(1)	Min of second coordinate	Pos;stat.min
c2max	Float	(1)	Max of second coordinate	Pos;stat.max
c3min	Float	(1)	Min of third coordinate	Pos;stat.min
c3max	Float	(1)	Max of third coordinate	Pos;stat.max
c1_resol_min	Float	(1)	Min resolution in first coordinate	Pos.resolution;stat.min (not in list)
c1_resol_max	Float	(1)	Max resolution in first coordinate	pos.resolution;stat.max (not in list)
c2_resol_min	Float	(1)	Min resolution in second coordinate	pos.resolution;stat.min (not in list)
c2_resol_max	Float	(1)	Max resolution in second coordinate	pos.resolution;stat.max (not in list)
c3_resol_min	Float	(1)	Min resolution in third coordinate	pos.resolution;stat.min (not in list)
c3_resol_max	Float	(1)	Max resolution in third coordinate	pos.resolution;stat.max (not in list)
spatial_frame_type	String		Flavor of coordinate system, defines the nature of coordinates	meta.code.class;pos.frame
incidence_min	float		Min incidence angle (solar zenith angle)	pos.posAng;stat.min
incidence_max	float		Max incidence angle (solar zenith angle)	pos.posAng;stat.max
emergence_min	float		Min emergence angle	pos.posAng;stat.min
emergence_max	float		Max emergence angle	pos.posAng;stat.max
phase_min	float		Min phase angle	pos.phaseAng;stat.min (TBC)
phase_max	String		Max phase angle	pos.phaseAng;stat.max (TBC)
instrument_host_name	String		Standard name of the observatory or spacecraft	meta.id;instr.obsty
instrument_name	String		Standard name of instrument	meta.id;instr
measurement_type	String		UCD(s) defining the data	meta.ucd
Optional				

parameters				
access_url	String		URL of the data file, case sensitive	meta.ref.url;meta.file
access_format	String		File format type	meta.code.mime?
access_estsize	Integer	kB	Estimate file size in kB	phys.size;meta.file (TBC)
preview_url	Integer		URL of a preview image	meta.ref.url;meta.file
native_access_url	String		URL of the data file in native form, case sensitive	meta.ref.url;meta.file
native_access_format	String		File format type in native form	meta.code.mime?
file_name	String		Name of the data file only, case sensitive	meta.id;meta.file
species	String		Identifies a chemical species, case sensitive	meta.id;phys.atmol
element_name	String		Secondary name (can be standard name of region of interest)	meta.id;pos
reference	String		Bibcode or other bilbio id, URL…	meta.bib
ra	Float		Right ascension	pos.eq.ra;meta.main
dec	Float		Declination	pos.eq.dec;meta.main
solar_longitude	Float		Solar longitude Ls	pos.posAngle (TBC)
local_time_min	Float	h	Local time at observed region	time.phase;stat.min?
local_time_max	Float	h	Local time at observed region	time.phase;stat.max?
target_distance	Float	km	Observer-target distance	pos.distance
particle_spectral_type	String			
particle_spectral_range_min	Float			
particle_spectral_range_max	Float			
particle_spectral_sampling_step_min	Float			
particle_spectral_sampling_step_max	Float			
particle_spectral_resolution_min	Float			spect.resolution;stat.min
particle_spectral_resolution_max	Float			spect.resolution;stat.max
Relative to service / Table header				
processing_level	Integer		CODMAC calibration level	meta.code;obs.calib
publisher	String		Resource publisher	meta.name
reference	String		Reference publication (bibcode, doi,	meta.bib

			URL…)	
service_title	String		Title of resource	meta.title
spatial_coordinate_description	String		?	?
spatial_origin	String		Defines the frame origin	meta.ref;pos.frame
time_origin	String		Defines where the time is measured	?
target_region	String		Type of region of interest	meta.id;class

(1) depending on context (as given by spatial_frame_type)

Appendix D: Parameter description

The xml definition below provides a standard description of EPN-TAP main parameters. It must appear in the service response and must therefore be included in the service definition, e.g. in the q.rd file used by a DaCHS server.

```
<table id="epn_core" onDisk="True" adql="True">
 [...]
 <column name="index" type="integer" required="True"
     ucd="meta.id"
     description="Identifies granule, provides direct access"/>
 <column name="resource_type" type="text"
     ucd="meta.code.class"
     description="Either dataset or granule"/>
 <column name="dataset_id" type="text"
     ucd="meta.id"
     description="Links granules to a dataset"/>
 <column name="dataproduct_type"  type="text"
     ucd="meta.code.class"
     description="Organization of the data product (from enumerated list)"/>
 <column name="target_name"  type="text"
     ucd="meta.id;src"
     description="Name of target (IAU standard)"/>
 <column name="target_class"  type="text"
     ucd="meta.code.class;src"
     description="Type of target, from enumerated list"/>
 <column name="target_distance"  type="double precision"
     ucd="pos.distance" unit="km"
     description="Mean distance from observer"/>
 <column name="time_min"  type="double precision"
     ucd="time.start" unit="d"
     description="Acquisition start time (in JD)"/>
 <column name="time_max" type="double precision"
     ucd="time.end" unit="d"
     description="Acquisition stop time (in JD)"/>
 <column name="time_sampling_step_min"   type="double precision"
    ucd="time.interval;stat.min"  unit="s"
     description="Min time sampling step"/>
 <column name="time_sampling_step_max"   type="double precision"
     ucd="time.interval;stat.max"  unit="s"
     description="Max time sampling step"/>
 <column name="time_exp_min"   type="double precision"
     ucd="time.duration;stat.min"  unit="s"
     description="Min integration time"/>
 <column name="time_exp_max"   type="double precision"
     ucd="time.duration;stat.max"  unit="s"
     description="Max integration time"/>
 <column name="spectral_range_min"   type="double precision"
     ucd="em.freq;stat.min" unit="Hz"
     description="Min spectral range (as frequency)"/>
 <column name="spectral_range_max"   type="double precision"
     ucd="em.freq;stat.max" unit="Hz"
     description="Max spectral range (as frequency)"/>
```

```
<column name="spectral_sampling_step_min"  type="double precision"
    ucd="em.freq.step;stat.min" unit="Hz"
    description="Min spectral sampling step (as frequency)"/>
<column name="spectral_sampling_step_max"  type="double precision"
      ucd="em.freq.step;stat.max" unit="Hz"
    description="Max spectral sampling step (as frequency)"/>
<column name="spectral_resolution_min"  type="double precision"
    ucd="spect.resolution;stat.min" unit="Hz"
    description="Min spectral resolution (as frequency)"/>
<column name="spectral_resolution_max"  type="double precision"
    ucd="spect.resolution;stat.max" unit="Hz"
    description="Max spectral resolution (as frequency)"/>
<column name="c1min"   type="double precision"
    ucd="pos;stat.min" unit="deg"
    description="Min (westernmost) longitude on planetary surface"/>
<column name="c1max"   type="double precision"
    ucd="pos;stat.max" unit="deg"
    description="Max (easternmost) longitude on planetary surface"/>
<column name="c2min"   type="double precision"
    ucd="pos;stat.min" unit="deg"
    description="Min latitude on planetary surface"/>
<column name="c2max"   type="double precision"
    ucd="pos;stat.max" unit="deg"
    description="Max latitude on planetary surface"/>
<column name="c3min"   type="double precision"
    ucd="pos;stat.min" unit=""
    description="Coordinate not used (altitude or depth)"/>
<column name="c3max"   type="double precision"
    ucd="pos;stat.max" unit=""
    description="Coordinate not used (altitude or depth)"/>
<column name="c1_resol_min"   type="double precision"
    ucd="pos.resolution;stat.min" unit="deg"
    description="Min resolution in longitude"/>
<column name="c1_resol_max"   type="double precision"
    ucd="pos.resolution;stat.max" unit="deg"
    description="Max resolution in longitude"/>
<column name="c2_resol_min"   type="double precision"
    ucd="pos.resolution;stat.min" unit="deg"
    description="Min resolution in latitude"/>
<column name="c2_resol_max"   type="double precision"
    ucd="pos.resolution;stat.max" unit="deg"
    description="Max resolution in latitude"/>
<column name="c3_resol_min"   type="double precision"
    ucd="pos.resolution;stat.min" unit=""
    description="not used"/>
<column name="c3_resol_max"   type="double precision"
    ucd="pos.resolution;stat.max" unit=""
    description="not used"/>
<column name="spatial_frame_type"  type="text"
    ucd="meta.code.class;pos.frame"
    description="Defines the nature of coordinates (from enumerated list)"/>
<column name="spatial_origin"  type="text"
    ucd="meta.code"
    description="Spatial origin of coordinate system"/>
<column name="local_time_min"   type="double precision"  unit="h"
```

```
    ucd="time.phase;stat.min"
    description="Min local time (angle to the solar meridian / 24)"/>
 <column name="local_time_max"   type="double precision"  unit="h"
    ucd="time.phase;stat.max"
    description="Max local time (angle to the solar meridian / 24)"/>
 <column name="incidence_min"    type="double precision"
    ucd="pos.posAng;stat.min" unit="deg"
    description="Min incidence angle (solar zenith angle)"/>
 <column name="incidence_max"    type="double precision"
    ucd="pos.posAng;stat.max" unit="deg"
    description="Max incidence angle (solar zenith angle) "/>
 <column name="emergence_min"    type="double precision"
    ucd="pos.posAng;stat.min" unit="deg"
    description="Min emergence angle"/>
 <column name="emergence_max"    type="double precision"
    ucd="pos.posAng;stat.max" unit="deg"
    description="Max emergence angle"/>
 <column name="phase_min"    type="double precision"
    ucd="pos.phaseAng;stat.min" unit="deg"
    description="Min phase angle"/>
 <column name="phase_max"   type="double precision"
    ucd="pos.phaseAng;stat.max" unit="deg"
    description="Max phase angle"/>
 <column name="instrument_host_name"  type="text"
    ucd="meta.id;instr.obsty"
    description="Standard name of the observatory or spacecraft"/>
 <column name="instrument_name"  type="text"
    ucd="meta.id;instr"
    description="Standard name of the instrument"/>
 <column name="measurement_type"  type="text"
    ucd="meta.ucd"
    description="UCD(s) defining the nature of measurements"/>
 <column name="access_url"  type="text"
    ucd="meta.ref.url"
    description="URL of the data file"/>
 <column name="access_format"  type="text"
    ucd="meta.code;mime"
    description="File format type"/>
 <column name="access_estsize"  type="integer" unit="kbyte" required="True"
    ucd="phys.size;meta.file"
    description="Estimate file size in kB"/>
 <column name="file_name"  type="text"
    ucd="meta.id;meta.file"
    description="Name of the data file"/>
<column name="reference"  type="text"
    ucd="meta.bib"
    description="Extra: bibliograpic reference"/>
 <column name="processing_level"  type="integer" required="True"
    ucd="meta.code;obs.calib"
    description="Level of calibration (CODMAC level)"/>
 [...]
</table>
```

Appendix E: Reserved keywords

Some keywords are used by the languages involved in the VO mechanism and must not be used as parameter names.
This includes ADQL keywords:

ABS	ACOS	AREA	ASIN	ATAN	ATAN2	BOX	CEILING	CENTROID
CIRCLE	CONTAINS	COORD1	COORD2	COORDSYS	COS	DEGREES	DISTANCE	EXP
FLOOR	INTERSECTS	LOG	LOG10	MOD	PI	POINT	POLYGON	POWER
RADIANS	REGION	RAND	ROUND	SIN	SQRT	TOP	TAN	TRUNCATE

plus SQL92 keywords:

ABSOLUTE	ACTION	ADD	ALL	ALLOCATE	ALTER
AND	ANY	ARE	AS	ASC	ASSERTION
AT	AUTHORIZATION	AVG	BEGIN	BETWEEN	BIT
BIT_LENGTH	BOTH	BY	CASCADE	CASCADED	CASE
CAST	CATALOG	CHAR	CHARACTER	CHARACTER_LENGTH	CHAR_LENGTH
CHECK	CLOSE	COALESCE	COLLATE	COLLATION	COLUMN
COMMIT	CONNECT	CONNECTION	CONSTRAINT	CONSTRAINTS	CONTINUE
CONVERT	CORRESPONDING	COUNT	CREATE	CROSS	CURRENT
CURRENT_DATE	CURRENT_TIME	CURRENT_TIMESTAMP	CURRENT_USER	CURSOR	DATE
DAY	DEALLOCATE	DECIMAL	DECLARE	DEFAULT	DEFERRABLE
DEFERRED	DELETE	DESC	DESCRIBE	DESCRIPTOR	DIAGNOSTICS
DISCONNECT	DISTINCT	DOMAIN	DOUBLE	DROP	ELSE
END	END-EXEC	ESCAPE	EXCEPT	EXCEPTION	EXEC
EXECUTE	EXISTS	EXTERNAL	EXTRACT	FALSE	FETCH
FIRST	FLOAT	FOR	FOREIGN	FOUND	FROM
FULL	GET	GLOBAL	GO	GOTO	GRANT
GROUP	HAVING	HOUR	IDENTITY	IMMEDIATE	IN
INDICATOR	INITIALLY	INNER	INPUT	INSENSITIVE	INSERT
INT	INTEGER	INTERSECT	INTERVAL	INTO	IS
ISOLATION	JOIN	KEY	LANGUAGE	LAST	LEADING
LEFT	LEVEL	LIKE	LOCAL	LOWER	MATCH
MAX	MIN	MINUTE	MODULE	MONTH	NAMES
NATIONAL	NATURAL	NCHAR	NEXT	NO	NOT
NULL	NULLIF	NUMERIC	OCTET_LENGTH	OF	ON
ONLY	OPEN	OPTION	OR	ORDER	OUTER
OUTPUT	OVERLAPS	PAD	PARTIAL	POSITION	PRECISION
PREPARE	PRESERVE	PRIMARY	PRIOR	PRIVILEGES	PROCEDURE
PUBLIC	READ	REAL	REFERENCES	RELATIVE	RESTRICT
REVOKE	RIGHT	ROLLBACK	ROWS	SCHEMA	SCROLL
SECOND	SECTION	SELECT	SESSION	SESSION_USER	SET
SIZE	SMALLINT	SOME	SPACE	SQL	SQLCODE
SQLERROR	SQLSTATE	SUBSTRING	SUM	SYSTEM_USER	TABLE
TEMPORARY	THEN	TIME	TIMESTAMP	TIMEZONE_HOUR	TIMEZONE_MINUTE
TO	TRAILING	TRANSACTION	TRANSLATE	TRANSLATION	TRIM
TRUE	UNION	UNIQUE	UNKNOWN	UPDATE	UPPER
USAGE	USER	USING	VALUE	VALUES	VARCHAR
VARYING	VIEW	WHEN	WHENEVER	WHERE	WITH
WORK	WRITE	YEAR	ZONE		

Appendix F: MIME-types

The values listed below are MIME-types used with the access_format parameter to define the type of the files under access_url.

MIME-type	Shortname	Extension	Definition	Datamodel
image/jpeg	jpeg	.jpg, .jpeg	A 2D JPEG graphic image (likewise for GIF, PNG, etc.)	obscore
image/png	png	.png	A digital format for still images. Portable Network Graphics (PNG)	spase
image/tiff	tiff	.tif, .tiff	A binary format for still pictures. Tagged Image Format File (TIFF). Originally developed by Aldus and now controlled by Adobe.	spase
Image/gif	gif	.gif	Graphic Interchange Format (GIF) first introduced in 1987 by CompuServe. GIF uses LZW compression and images are limited to 256 colours.	spase
application/x-votable+xml	votable	.xml, .vot	Any generic VOTable file	obscore
text/csv	csv	.csv	Tabular data in comma separated values format	obscore
text/html	html	.htm, .html	Text in HTML format	obscore
text/plain	txt	.txt, .asc	Any generic text file	obscore
text/tab-separated-values	tsv	.tsv	Tabular data in tab separated values format	obscore
text/xml	xml	.xml	Any generic XML file	obscore
image/fits	fits	.fit, .fits	Any multidimensional regularly sampled FITS image or cube	obscore
application/x-fits-mef	mef		A FITS multi-extension file (multiple extensions)	obscore
image/x-fits-gzip	fits	.gz	A GZIP-compressed FITS image	obscore
image/x-fits-hcompress	fits		A FITS image using HCOMPRESS compression	obscore
application/x-fits-bintable	bintable	.fit, .fits	A FITS binary table (single BINTABLE extension)	obscore
application/x-fits-euro3d	euro3d		A FITS file in Euro3D format (multiobject spectroscopy)	obscore
application/fits	fits	.fit, .fits	Any generic FITS file	obscore
application/gml+xml	gml	.gml, .xml	Geography Markup Language (GML)	ogc
application/json	json		JavaScript Object Notation (JSON) file	epncore
application/octet-stream	bin	.bin, .dat	Binary Data	spase
application/octet-stream	idl	.idl, .sav	Interactive Data Language (IDL) save set. IDL is a proprietary format.	spase
application/octet-stream	matlab4	.mat	MATLAB Workspace save set, version 4. MAT-files are double-precision, binary, MATLAB format files. MATLAB is a proprietary product of The MathWorks.	spase
application/octet-stream	matlab6	.mat	MATLAB Workspace save set, version 6. MAT-files are double-precision, binary, MATLAB format files. MATLAB is a proprietary product of The MathWorks.	spase
application/octet-stream	matlab7	.mat	MATLAB Workspace save set, version 7. MAT-files are double-precision, binary, MATLAB format files. Version 7 includes data compression and Unicode encoding. MATLAB is a proprietary product of The MathWorks.	spase
application/pdf	pdf	.pdf	Any PDF file	obscore
application/postscript	ps	.ps, .eps, .ai	A page description programming language created by Adobe Systems Inc. that is a device-independent industry standard for representing text and graphics.	spase
application/vnd.geo+json	geojson	.json	GIS file format in json	ogc
application/vnd.google-earth.kml+xml	kml	.kml	Keyhole Markup Language	epncore
application/vnd.google-earth.kmz	kmz	.kmz	Keyhole Markup Language in a zipped file	epncore
application/vnd.ms-excel	xls	.xls, .xlsx	A Microsoft spreadsheet format used to hold a variety of data in tables which can include calculations.	spase
application/x-asdm	asdm		ALMA science data model (final export format still TBD)	obscore

application/x-cdf-istp	cdf	.cdf	Common Data Format (CDF) file compliant with ISTP	epncore
application/x-cdf-pds4	cdf	.cdf	Common Data Format (CDF) file compliant with PDS4	epncore
application/x-cef1	cef1	.cef	Cluster Exchange Format (CEF), version 1, is a self-documenting ASCII format designed for the exchange of data. The metadata contains information compatible with the ISTP recommendations for CDF.	spase
application/x-cef2	cef2	.cef	Cluster Exchange Format (CEF), version 2, is a self-documenting ASCII format designed for the exchange of data and introduced for Cluster Active Archive. Compared to version 1, the metadata description of vectors and tensors is different.	spase
application/x-directory	dir		Multiple files archive returned as a text list	obscore
application/x-geotiff	geotiff	.tiff	GeoTIFF is a public domain metadata standard which allows georeferencing information to be embedded within a TIFF file. The potential additional information includes map projection, coordinate systems, ellipsoids, datums, and everything else necessary to establish the exact spatial reference for the file. The GeoTIFF format is fully compliant with TIFF 6.0, so software incapable of reading and interpreting the specialized metadata will still be able to open a GeoTIFF format file.	ogc
application/x-hdf	hdf	.hf4	Hierarchical Data Format 4	epncore
application/x-hdf	hdf	.hf5	Hierarchical Data Format 5	epncore
application/x-netcdf	nc	.nc	Network Common Data Format (NetCDF) file version 3	epncore
application/x-netcdf4	nc	.nc4, .nc	Network Common Data Format (NetCDF) file version 4	epncore
application/x-tar	tar	.tar	Multiple files archive in TAR format	obscore
application/x-tar-gzip	gtar	.tgz	A GZIP-compressed TAR file (x-gtar also sometimes used	obscore
application/zip	zip	.zip	Multiple files archive in ZIP format	obscore
video/mpeg	mpeg	.mpeg, .mpg	A digital format for movies defined by the Motion Picture Experts Group	spase
video/quicktime	mov	.mov, .qt	A format for digital movies, as defined by Apple Computer.	spase
video/x-msvideo	avi	.avi	Audio Video Interleave (AVI)	spase
	idfs		*Instrument Data File Set (IDFS) is a set of files written in a prescribed format which contain data, timing data, and meta-data. IDFS was developed at Southwest Research Institute (SwRI).*	*spase*
	ncar		*The National Center for Atmospheric Research (NCAR) format. A complete description of that standard is given in appendix C of the "Report on Establishment & Operation of the Incoherent- Scatter Data Base", dated August 23, 1984, obtainable from NCAR, P.O. Box 3000 Boulder, Colorado 80307-3000.*	*spase*
	udf		*Universal Data Format (UDF). The Optical Technology Storage Association's Universal Disk Format, based on ISO 13346. See <http://www.osta.org/specs/index.htm>*	*spase*