EPN-TAP V2.0 parameters

List of EPN-TAP parameters
v2, 3/8/2015 (SE)
v3, 29/8/2015 (SE, integrating comments by PLS and BC)
v4, 7/9/2015 (SE, integrating comments by PLS)
v5, 7/10/2015 (SE, integrating comments by PLS: time_scale UCD)
v6, 7/11/2015 (SE, integrating comments by PLS/BC)
v7, 14/12/2015 (SE, added point 8, clarification of an older idea)
v8, 24/12/2015 (SE, review/integration of comments/corrections)
v9, 5-7/01/2016 (SE/BC, including comments from Chiara Marmo and Michel Gangloff)
v10, 21/02/2016 (SE, from model implementation of IKS)
v11, 11/5/2016 (SE, systematic comparison with ObsCore again & variations for spatial coordinates)
v12, 15/8/2016 (SE, minor fixes from service implementation)
2016-17 (SE/PLS: various updates, in particular UCD of coordinates and some optional parameters)
v14, 11/8/2017 (SE, added Utypes from current epntapv2 mixin + ObsCore v1.1)

EPN-TAP

- EPN-TAP is a VO access protocol dedicated to Planetary Science data. It is based on the TAP mechanism from IVOA, completed with sets of parameters and associated lists of values. In this regard, it is similar to ObsTAP but with a different scope.
- EPN-TAP was described in its first version in Erard et al, Astronomy and Computing 7 p.52-61 (2014). Version 1 was designed as part of the IDIS activity in Europlanet-RI (FP7).
- EPN-TAP version 2 is a major update of the protocol to accommodate larger services and simplify setup and use of data services. All parameters are described here.

New in v2

- The "laundry list" format makes services easier to design and to query
- Allows grouping of results from several services at once
- Supports multispacecraft observations
- Speeds up mirroring of services (support for partial updates)
- Better support of footprints, and better interface with GIS

Main evolutions relative to v1

1) The previous notion of dataset is deprecated

This was complex to handle in the database, and in general not relevant for the services.

2) Grouping of products is rationalized in version 2

A granule is still a record/line in the epn_core view database, and corresponds to the smallest data unit described by the service.
A "product" is typically a data file, or a service output, that can be reached through a URL
In version 2 both concepts coincide, while in v1 a single granule could be composed of several products related to the same initial data (an "observation" in v2)

In EPN-TAP v2, granules are referenced by 3 parameters:

- granule_uid provides a unique ID for the granule in the service, ie for each line in the epn_core view. It is equivalent to the previous index parameter (index is a reserved word in many database languages and should not have been used in the first place)
- granule_gid is related to a type of product: it is identical for all granules containing the same type of information for different observations (e.g., calibrated files). An explicit string is recommended in this field, with suggested standard values (e.g., preview/native/calibrated/geometry/projection…).
- obs_id is related to an observation: it is identical for all granules related to the same observation, containing different type of data (e.g.: raw and calibrated data, associated geometry, etc). In many EPN-TAP v1 services, such products were described together on the same line with a unique index parameter.

These 3 parameters can be arbitrary alphanumeric strings — see example application to APIS service below.

In practice, different products related to the same observation are no longer described together on a single line of the epn_core view, but on successive lines associated by the same obs_id, each with a different granule_gid (and a specific granule_uid, see Table 2 below).

Each line in the epn_core view must describe only one product (plus a thumbnail wherever relevant). The notion of "main product" (which was more or less explicit in v1) is therefore deprecated, and the epn_core view in v2 includes more lines than in v1. Although less compact that the previous table presentation, this list presentation is much more efficient for machine-handling, and easier to design.

3) The notion of table/service parameters is deprecated

Such parameters were available in the registry only, and were not directly accessible by TAP. In v2, constant parameters must be replicated in every line of the epn_core view.
Such parameters may be duplicated in the registry declaration though, so as to provide a fast description of services.

4) Footprints can be provided through s_region

s_region is a parameter in the ObsCore standard of IVOA, and ADQL allows for powerful query functions such as intersections or inclusions. s_region introduces a pgSphere variable of type spoly providing description of the observed area on a sphere, which allows for accurate searches.

This can be used to provide footprints of spatially extended data either on the sky or on a planetary surface (as long as it is reasonably ellipsoidal).
Use with 3D shape models needs study (this may fail with concave 3D shapes such as 67P). Also need to study how concave footprints/polygons are handled, TBC. The use in the context of GIS also has to be studied.

The C1/C2 min/max parameters can still be used to provide a bounding box of the observation.

5) Better support of evolving services

creation_date and modification_date are now mandatory parameters for every granule. The latter is intended to optimize mirroring of services, by identifying the granules to be updated/copied.

These parameters must be provided as ISO 8601 strings with format "2013-11-17T10:41:00.00+01:00" (with no space) where the indication of time zone is mandatory. The associated data type must be "timestamp". ADQL supports this format, and filtering based on time zone is possible.

6) Support of coordinated observations

The target_time_min/max parameters now provide observation time in the reference frame of the target. This is intended to facilitate the cross-correlation of observations from different locations, e.g., telescopic observations in support of space missions, or multi-spacecraft campaigns.

7) Axes ranges

All parameters defining a range are now introduced with a min and max value.
All floating point parameters are now in double precision to prevent errors.

8) Thematic extensions

Some science fields will require optional parameters, which need to be used consistently between services addressing the same field. Such extensions have to be designed by sub-groups involved in the corresponding data services, either as providers or users. This includes:

- Lab spectroscopy: parameters to describe mineralogical samples (and possibly other samples)
- Orbital/rotational parameters and physical properties of Solar System bodies
- Results of planetary 3D modelling run
- Exoplanets / planetary systems properties
- Bibliographic entries? May be manageable otherwise, through bibcode / doi interpretation

9) Parameters which must be provided

are now clearly identified - those are not only mandatory, they also must provide a value.
They are mostly related to service description and granule identification.

For Utypes : see epntap mixin v2 for standard parameters (grab them from an existing service)

+ See notes below the table.

Name, v2	Must be filled?	Name, v1	SQL type	Unit / Format / Range	Description	UCD **Please comment!**	UCD in Obscore 1.1 (30/3 /2016 version)	Utype	Comments / UCD
EPNCore mandatory parameters (Must be present, possibly empty) New or modified relative to v1						Current value current but dubious or *undefined* possible alternative (*but undefined*)	— ? : closest sense _ : N/A in ObsCore	from epntap v2 mixin (aug 2017) equivalent /close in ObsCore doc 1.1	
granule_uid	Y	index	Text		Internal table row index Unique ID in data service, also in v2. Can be alphanum.	meta.id meta.id; meta.dataset	meta.id		
granule_gid	Y		Text		Common to granules of same type (e.g. same map projection, or geometry data products). Can be alphanum.	meta.id	meta.id		
obs_id	Y		Text		Associates granules derived from the same data (e.g. various representations / processing levels). Can be alphanum., may be the ID of original observation.	meta.id meta.id;obs	meta.id	obscore: DataID. observationID	
_		resource_type	Text		Can be dataset or granule	(meta.code. class)	_		
_		dataset_id	Text		Dataset identification & granule reference	(meta.id)	_		
dataproduct_type	Y	dataproduct_type	Text		Organization of the data product, from enumerated list	meta.code. class	meta.id	Epn. dataProductType obscore: ObsDataset. dataProductType	
target_name	Y (TBC)	target_name	Text		Standard IAU name of target (from a list related to target class), case sensitive	meta.id;src	meta.id;src	Epn. TargetName	
target_class	Y	target_class	Text		Type of target, from enumerated list	~~meta.code. class;src~~ src.class	src.class	Epn. TargetClass	
time_min		time_min	Double	d (date as JD)	Acquisition start time (in JD). UTC measured at time_origin location (default is observer's frame)	~~time.start~~ time.start;obs time.start; obs.exposure (if exposed?)	time.start; obs.exposure	Char. TimeAxis. Coverage. Bounds. Limits. Interval. StartTime TBD (from ObsCore)	

time_max		time_max	Double	d (date as JD)	Acquisition stop time (in JD). UTC measured at time_origin location (default is observer's frame)	~~time.end~~ time.end;obs time.end;obs.exposure (if exposed?)	time.end;obs.exposure	Char.TimeAxis.Coverage.Bounds.Limits.Interval.StopTime TBD (from ObsCore)	
time_sampling_step_min		time_sampling_step_min	Double	s	Min time sampling step	time.interval;stat.min	time.resolution — ?	Epn.Time.Time_sampling_step_min	
time_sampling_step_max		time_sampling_step_max	Double	s	Max time sampling step	time.interval;stat.max		Epn.Time.Time_sampling_step_max	
time_exp_min		time_exp_min	Double	s	Min integration time	time.duration;obs.exposure;stat.min	time.duration;obs.exposure	Epn.Time.Time_exp_min	
time_exp_max		time_exp_max	Double	s	Max integration time	time.duration;obs.exposure;stat.max		Epn.Time.Time_exp_max	
spectral_range_min		spectral_range_min	Double	Hz	Min spectral range (frequency)	em.freq;stat.min	em.wl;stat.min (always as wl)	Epn.Spectral.Spectral_range_min	Always as freq
spectral_range_max		spectral_range_max	Double	Hz	Max spectral range (frequency)	em.freq;stat.max	em.wl;stat.max	Epn.Spectral.Spectral_range_max	
spectral_sampling_step_min		spectral_sampling_step_min	Double	Hz	Min spectral sampling step	*em.freq.step;stat.min*	meta.number	Epn.Spectral.Spectral_sampling_step_min	
spectral_sampling_step_max		spectral_sampling_step_max	Double	Hz	Max spectral sampling step	*em.freq.step;stat.max*	meta.number	Epn.Spectral.Spectral_sampling_step_max	
spectral_resolution_min		spectral_resolution_min	Double	Hz	Min spectral resolution	spect.resolution;stat.min	spect.resolution	Epn.Spectral.Spectral_resolution_min	
spectral_resolution_max		spectral_resolution_max	Double	Hz	Max spectral resolution	spect.resolution;stat.max		Epn.Spectral.Spectral_resolution_max	
c1min		c1min	Double	(1) Longitude from 0. to 359.9999 RA from 0. to 23.9999	Min of first coordinate	pos;stat.min pos.distance;stat.min (bof) or *pos.radius;stat.min* (does not exist) for spherical & cylindrical pos.eq.ra;stat.min for celestial pos.bodyrc.lon;stat.min for body pos.cartesian.x;stat.min for Cartesian pos.healpix for healpix (with 2 parameters? - weird) - TBC empty ("") for none (and no unit)	pos.eq.ra	Epn.Spatial.Spatial_range.c1min	
c1max		c1max	Double	(1)	Max of first coordinate	pos;stat.max, etc		Epn.Spatial.Spatial_range.c1max	

c2min		c2min	Double	(1) Latitude from -89.9999 to +89.9999	Min of second coordinate	pos;stat.min pos.angDistance;stat.min or pos.az.zd;stat.min (for zenithal distance) for spherical or pos.az.azi;stat.min (for azimuth) for cylindrical pos.eq.dec;stat.min for celestial pos.bodyrc.lat;stat.min for body pos.cartesian.y;stat.min for Cartesian empty ("") for none (and no unit)	pos.eq.dec	Epn.Spatial.Spatial_range.c2min	
c2max		c2max	Double	(1)	Max of second coordinate	pos;stat.max, etc		Epn.Spatial.Spatial_range.c2max	
c3min		c3min	Double	(1)	Min of third coordinate	pos;stat.min pos.AngDistance;stat.min or pos.az.azi;stat.min (for azimuth) for spherical pos.distance;stat.min for cylindrical pos.distance;stat.min for celestial pos.bodyrc.alt;stat.min for body? (from surface only, implicitly from reference level) or pos.distance;pos.bodyrc;stat.min for body (from center)? pos.cartesian.z;stat.min for Cartesian empty ("") for none (and no unit)		Epn.Spatial.Spatial_range.c3min	
c3max		c3max	Double	(1)	Max of third coordinate	pos;stat.max, etc		Epn.Spatial.Spatial_range.c3max	

s_region			spoly	PgSphere spoly	ObsCore-like footprint, valid for celestial, spherical, or body-fixed frames.	phys.outline; obs.field (was initially instr.fov, to be corrected)	phys.outline; obs.field	obscore: Char. SpatialAxis. Coverage. Support.Area	ObsCore value recently updated (was phys. angArea;obs) Frame may be identified in q.rd (UNKNOWN Frame) Do we need another param for GIS interface?
c1_resol_min		c1_resol_min	Double	(1)	Min resolution in first coordinate	*pos. resolution; stat.min* pos. angResolution ;stat.min if angular	pos. angResolution;stat.min	Epn.Spatial. Spatial_resol ution. c1_resol_min	(UCD is deprecated, but needed!)
c1_resol_max		c1_resol_max	Double	(1)	Max resolution in first coordinate	*pos. resolution; stat.max*	pos. angResolution;stat.max	Epn.Spatial. Spatial_resol ution. c1_resol_max	(UCD is deprecated, but needed!)
c2_resol_min		c2_resol_min	Double	(1)	Min resolution in second coordinate	*pos. resolution; stat.min pos. angResolution ;stat.min if angular*		Epn.Spatial. Spatial_resol ution. c2_resol_min	(UCD is deprecated, but needed!)
c2_resol_max		c2_resol_max	Double	(1)	Max resolution in second coordinate	*pos. resolution; stat.max*		Epn.Spatial. Spatial_resol ution. c2_resol_max	(UCD is deprecated, but needed!)
c3_resol_min		c3_resol_min	Double	(1)	Min resolution in third coordinate	*pos. resolution; stat.min pos. angResolution ;stat.min if angular (spherical only)*		Epn.Spatial. Spatial_resol ution. c3_resol_min	(UCD is deprecated, but needed!)
c3_resol_max		c3_resol_max	Double	(1)	Max resolution in third coordinate	*pos. resolution; stat.max*		Epn.Spatial. Spatial_resol ution. c3_resol_max	(UCD is deprecated, but needed!)
spatial_frame _type	Y	spatial_frame _type	Text	(1) Use "none" if undefined	Flavor of coordinate system, defines the nature of coordinates. From enumerated list	meta.code. class;pos. frame	_		A value is required by DaCHS (query will return errors if empty)
incidence_min		incidence_min	Double	deg	Min incidence angle (solar zenithal angle)	pos.;stat.min *pos. incidenceAng ;stat.min* posAng	_	Epn. View_angle. Incidence_an gle_min	
incidence_m ax		incidence_m ax	Double	deg	Max incidence angle (solar zenithal angle)	pos.posAng; stat.max *pos. incidenceAng ;stat.max*	_	Epn. View_angle. Incidence_an gle_max	
emergence_ min		emergence_ min	Double	deg	Min emergence angle	pos.posAng; stat.min *pos. emergenceA ng;stat.min*	_	Epn. View_angle. Emergence_ angle_min	
emergence_ max		emergence_ max	Double	deg	Max emergence angle	pos.posAng; stat.max *pos. emergenceA ng;stat.max*	_	Epn. View_angle. Emergence_ angle_max	
phase_min		phase_min	Double	deg	Min phase angle	pos. phaseAng; stat.min		Epn. View_angle. Phase_angle _min	

phase_max		phase_max	Double	deg	Max phase angle	pos.phaseAng;stat.max		Epn.View_angle.Phase_angle_max	
instrument_host_name		instrument_host_name	Text		Standard name of the observatory or spacecraft	meta.id;instr.obsty	meta.id;instr.tel	Provenance.ObsConfig.Facility.name	
instrument_name		instrument_name	Text		Standard name of instrument	meta.id;instr	meta.id;instr	Provenance.ObsConfig.Instrument.name	
measurement_type		measurement_type	Text		UCD(s) defining the data	meta.ucd	meta.ucd	Epn.Measurement_type	
processing_level		processing_level	Integer		CODMAC calibration level in v1	~~meta.code; obs.calib~~ · meta.calibLevel	meta.code;obs.calib	~ obscore:ObsDataset.calibLevel	To be replaced by PDS4 values in v2?
creation_date	Y		Timestamp	(ISO-8601 String)	Date of first entry of this granule	time.creation	time;meta.dataset		
modification_date	Y		Timestamp	(ISO-8601 String)	Date of last modification (used to handle mirroring)	*time.update* time.processing? (not quite this)			
release_date	Y		Timestamp	(ISO-8601 String)	Start of public access period (set to creation_date if no proprietary period)	time.release	time.release	obscore:Curation.releaseDate	
service_title	Y	service_title (still "title" in many v1 services)	Text		Title of resource (an acronym really, will be used to handle multiservice results)	meta.title			
Optional parameters									
access_url		access_url	Text		URL of the data file, case sensitive. Can point to a script. If present, next 2 parameters must also be present.	meta.ref.url;meta.file	meta.ref.url	Obs.Access.Reference	Use this to link data! Can accommodate a datalink if access_format = 'application/x-votable+xml;content=datalink' (UCD from ObsCore)
access_format		access_format	Text	(mime type in lowercase)	File format type	meta.code.mime	meta.code.mime	Obs.Access.Format	
access_estsize		access_estsize	Integer	kbyte	Estimate file size in kbyte (with this spelling)	phys.size;meta.file	phys.size;meta.file	Obs.Access.Size	
data_access_url *(best avoided and replaced by datalink in access_url; TBC but this is what ObsCore does)*			Text		*If access_format indicates a detached label, this parameter is mandatory and points to the corresponding data file - both will be handled by the client before samping it to tools or downloading- TBC*	meta.ref.url;meta.file			

access_md5			Text		MD5 Hash for the file when available (real file, not script)	*meta.checksum; meta.file*			
_		preview_url	Integer		URL of a preview image (std format with adequate resolution for user's purpose)	(meta.ref.url; meta.file)			replaced by another granule group
_		native_access_url	Text		URL of the data file in native form, case sensitive	(meta.ref.url; meta.file)			replaced by another granule group
_		native_access_format	Text		File format type in native form	(meta.id; class) (or meta.code.mime if we use MIME type)			replaced by another granule group
thumbnail_url			Text		URL of a thumbnail image with predefined size (png ~200 pix, for use in a client only)	meta.ref.url; meta.file meta.ref.url; meta.preview			
file_name		file_name	Text		Name of the data file only, case sensitive	meta.id;meta.file	meta.title; obs — ?		ObsCore obs_title is for a short free text string describing the granule. Do we want this?
species		species	Text		Identifies a chemical species, case sensitive	meta.id;phys.atmol			
filter			Text		Identifies a filter in use (e.g. imaging)	meta.id;instr.filter			Informative only, free format (no list). Search can only rely on spectral range, as ObsCore does.
alt_target_name			Text		Provides alternative target name if more common (e.g. comets)	meta.id;src			
target_region		target_region	Text		Type of region of interest	~~meta.id;class~~ obs.field			
feature_name		element_name	Text		Secondary name (can be standard name of region of interest)	~~meta.id;pos~~ obs.field			
bib_reference		reference	Text		Bibcode preferred if available (does that include link?), doi, or other biblio id, URL…	meta.bib meta.bib.bibcode (if bibcode)	meta.bib.bibcode (always as bibcode)	obscore: Curation.reference	

internal_reference			Text		List of granule_uid(s) in the current service	meta.id.cross			Use to link one granule to a set of other granules. Only to solve situations that would otherwise require several tables
external_link			Text	(url)	Link to a web page providing more details on the *granule*.	meta.ref.url			Link to an individual page in a web site associated to the database, e.g., a planet page in Exoplanets service. This is a way to provide extra granule information which cannot be accommodated in the table.
subsolar_longitude			Double	deg	Sub-solar point	pos.bodyrc.lon			Provided in the most natural body-related coordinate frame, E-handed - seems to require 'body'
subsolar_latitude			Double	deg	Sub-solar point	pos.bodyrc.lat			_
subobserver_longitude			Double	deg	Sub-observer point (sub-Earth for ground based observations)	pos.bodyrc.lon			_
subobserver_latitude			Double	deg	Sub-observer point (sub-Earth for ground based observations)	pos.bodyrc.lat			_
ra		ra	Double	deg only (like ObsCore)	Right ascension (not hour angle!)	pos.eq.ra; meta.main			
dec		dec	Double	deg	Declination	pos.eq.dec; meta.main			
radial_distance			Double	km	Distance from center (in body-fixed frame)	pos.distance; pos.bodyrc			
altitude_from shape			Double	km	Altitude above shape model / DTM (in body-fixed frame)	pos.bodyrc.alt			
solar_longitude_min		solar_longitude	Double	deg	Min Solar longitude Ls (location on orbit / season)	pos.posang pos.bodyrc.lon ;pos.heliocentric; stat.min (posang means something else)			

solar_longitude_max		solar_longitude	Double	deg	Max Solar longitude Ls (location on orbit / season)	pos.posang pos.bodyrc.lon;pos.heliocentric; stat.max (posang means something else)			
local_time_min		local_time_min	Double	h	Local time at observed region	~~time.phase; stat.min~~ time.period.rotation;time.phase;stat.min			
local_time_max		local_time_max	Double	h	Local time at observed region	~~time.phase; stat.max~~ time.period.rotation;time.phase;stat.max			
target_distance_min		target_distance (no min/max)	Double	km	Observer-target distance	pos.distance; stat.min			
target_distance_max			Double	km	Observer-target distance	pos.distance; stat.max			
target_time_min			Double	d	Observing time in target frame	time.start			(simplest way to look for coordinated observations)
target_time_max			Double	d		time.end			
earth_distance_min			Double	au	Earth-target distance	pos.distance; stat.min			
earth_distance_max			Double	au		pos.distance; stat.max			
sun_distance_min			Double	au	Sun-target distance	pos.distance; stat.min			
sun_distance_max			Double	au		pos.distance; stat.max			
particle_spectral_type		particle_spectral_type	Text			Use phys.particle?			Particle spectroscopy extension
particle_spectral_range_min		particle_spectral_range_min	Double			Use phys.particle?			
particle_spectral_range_max		particle_spectral_range_max	Double			Use phys.particle?			
particle_spectral_sampling_step_min		particle_spectral_sampling_step_min	Double			Use phys.particle?			
particle_spectral_sampling_step_max		particle_spectral_sampling_step_max	Double			Use phys.particle?			
particle_spectral_resolution_min		particle_spectral_resolution_min	Double			spect.resolution; stat.min Use phys.particle?			
particle_spectral_resolution_max		particle_spectral_resolution_max	Double			spect.resolution; stat.max Use phys.particle?			
mass			Double	kg		phys.mass			Solar System Objects extension (generic values in catalogues, not observations)
sideral_rotation_period			Double	h		time.period.rotation			
mean_radius			Double	km		phys.size.radius			
equatorial_radius			Double	km		phys.size.radius			

polar_radius			Double	km		phys.size.radius			
Other optional parameters		These parameters were "Relative to service" (i.e., included in Table header) in v1; they are now regular optional parameters located in the table.							
publisher		publisher	Text		Resource publisher	meta.name meta.curation	meta.ref.url; meta.curation — ?	~ obscore: Curation.publisherID	
spatial_coordinate_description		spatial_coordinate_description	Text		ID of specific coordinate system and version / properties (~COOSYS)	meta.code.class;pos.frame			still TBD. Discussion in progress here: EPN-TAP v2: Current discussion topic
spatial_origin		spatial_origin	Text		Defines the frame origin	meta.ref;pos.frame			
time_origin		time_origin	Text		Defines *where* the time is measured (e. g., ground vs spacecraft) [target_time is of course always on target].	meta.ref;time.scale or pos;time.scale			TBC
time_scale			Text		Always UTC in *data* services (may be relaxed in computational services such as ephemeris) - from enumerated list	time.scale			

(1): depending on context (as given by spatial_frame_type). Please comment here: EPN-TAP v2: Current discussion topic

Beware that datatypes apply to the epn_core view, not to the q.rd file where they can be different

Example table:

File name-type	granule_uid	granule_gid	obs_id
A-Raw	1	native	A
A-Calib	2	calibrated	A
A-geom	3	geometry	A
A-proj	4	projected	A
B-Raw	5	native	B
B-Calib	6	calibrated	B
B-geom	7	geometry	B
B-proj	8	projected	B

Other modifications

• **multivalued lists** = first entry#second entry#…#last entry, or scalar (with no #)
Values separator = #
No quotes around the list

This can be parsed by ADQL/RegTAP function ivo_hashlist_has like this:
*select * from vvex.epn_core where 1 = ivo_hashlist_has(lower(target_name),'Venus')*

Where the *lower* function is mandatory to handle values possibly containing upper cases (this is implicit on the 2nd argument)

Beware that only complete elements between separators will be found. The provider has to split the string according to expected searches, e.g.:
Composite Infrared Spectrometer#CIRS
not ~~*Composite Infrared Spectrometer (CIRS)*~~

Parameters supporting multivalued lists include:
dataproduct_type *(this one is best avoided when possible)*
target_name
alt_target_name
target_class
instrument_host_name
instrument_name
measurement_type
bib_reference
processing_level

• **Defaut values** (to be reviewed):
NULL/void: will never return an answer to a query using this parameter (TBC, seems ADQL-related. To be corrected if it is a limitation of the client)
For float / double : -inf for *_min +inf for *_max – still TBC (NaN won't do). To be tested on a real case.
Not needed for strings? (i.e. NULL/void is OK?)
Default behavior looks OK in fact. If we need to preserve the NULL values on queried parameters (ie, when NULL is considered as "I don't know"), this can be done with OR param = NULL (on option in the client? This is feasible manually on TapHandle)

• **UCDs**: to be reviewed against PDS4 and IPDA, and completed (ObsCore checked)

• **Processing levels**: to be reviewed against PDS4 (again- mostly related to geometry, considered as derived in PDS4)

• **min vs max**:
if only one value available, it must appear in both fields

• **Optional parameters**: they come in sets that are logically related; if one is present, the related ones must be present also (e.g., 3 access_* parameters)

• **Granule_gid**: any general indication to providers? I.e.: preview, native, calibrated, geometry…
A client should be able to display the values present in a service, TBC

• Reshuffle previous "service parameters":

- Mandatory :
  processing_level -mandatory
  service_title –mandatory
  publisher -mandatory???
- Optional - TBC
  spatial_coordinate_description (default = body-fixed or J2000)
  spatial_origin (default = body center or SS barycenter? Or observer location)
  time_origin (default = observer)
  time_scale (default = UTC – no other values allowed in *data* services? [only in computational services, e.g. ephemeris])
  Same values to be used in registry declaration

• **Support for PDS3 detached labels** (proposal)

access_format = "PDS3label" (it has to be here: we need to know that there is a detached label)
then access_url points to the label, and data_access_url points to the file (param is mandatory in this case - although the data file name is in the label, it can be in another directory)
A script can then recover both files and do something with them. This mechanism can be extended to other formats with detached labels (ENVI…).
Solution with datalink seems OK: detached labels provided under datalink_url in a table - although no attempt made to read them from the portal, yet.

• **Utypes**

Need to clean up current doc (2.0). Utypes are = DM fields. They are supposedly used to identify the meaning of parameters and help e.g. tools to grab required quantities - This will not work in some areas though, e. g. with spectral tools as they currently use UCD instead of Utype for this purpose (not many tools appear to actually rely on Utype in fact). See discussion here for usage (a bit old?): http://www.ivoa.net/documents/Notes/UTypesUsage/20130213/NOTE-utypes-usage-1.0-20130213.html

To handle this in practice:

- Associate each parameter to a specific Utype in EPNCore - all names need to start with the epncore: prefix/namespace.
- Then map epncore Utype to other DM (find equivalent parameter, or trace back the original templates of EPNCore parameters - often from ObsCore)
- Reuse Utype from other DM each times it makes sense - TBC: the epncore: namespace is still required (even when using Utypes from other DM)
  This allows tools to handle EPNCore parameters like existing parameters from other DM, i.e., with no specific implementation
  Pb is that small differences in the use of parameters may preclude reusing the same Utype (TBC: does that applies to units also?)
- Cross our fingers: known Utype (from other DMs) may be usable in existing tools (e.g. a tool supporting Provenance would grab equivalent info in EPN-TAP services transparently)
  Unclear if the use of the namespace makes it more complicated in tools.

Example of V1 to V2 conversion with APIS database:

EPNcore Table v1

(was not really compliant…)

index	resource_type	dataset_id	access_url	access_format	preview_url	native_access_url	native_access_format
23801	granule	original_data	o5g202x4q_x2d.jpg	jpg	o5g202x4q_x2d_small.jpg	o5g202x4q_x2d.fits	fits
23802	granule	processed_data	o5g202x4q_proc.jpg	jpg	o5g202x4q_proc_small.jpg	o5g202x4q_proc.fits	fits
23803	granule	cylindric_proj	o5g202x4q_cyl.jpg	jpg	o5g202x4q_cyl_small.jpg		
23804	granule	polar_proj_north	o5g202x4q_pol_n.jpg	jpg	o5g202x4q_pol_n_small.jpg		
23805	granule	polar_proj_south	o5g202x4q_pol_s.jpg	jpg	o5g202x4q_pol_s_small.jpg		

EPNcore Table v2

granule_uid	granule_gid	obs_id	access_url	access_format	thumbnail_url
o5g202x4q_x2d	original_data	o5g202x4q	o5g202x4q_x2d.fits	image/fits	o5g202x4q_x2d_small.jpg
o5g202x4q_x2d_prev	original_data_preview	o5g202x4q	o5g202x4q_x2d.jpg	image/jpg	o5g202x4q_x2d_small.jpg
o5g202x4q_proc	processed_data	o5g202x4q	o5g202x4q_proc.fits	image/fits	o5g202x4q_proc_small.jpg
o5g202x4q_proc_prev	processed_data_preview	o5g202x4q	o5g202x4q_proc.jpg	image/jpg	o5g202x4q_proc_small.jpg
o5g202x4q_cyl	cylindrical_projection	o5g202x4q	o5g202x4q_cyl.jpg	image/jpg	o5g202x4q_cyl_small.jpg
o5g202x4q_pol_n	polar_projection_north	o5g202x4q	o5g202x4q_pol_n.jpg	image/jpg	o5g202x4q_pol_n_small.jpg
o5g202x4q_pol_s	polar_projection_south	o5g202x4q	o5g202x4q_pol_s.jpg	image/jpg	o5g202x4q_pol_s_small.jpg