Setting up an EPN-TAP service

Work in Progress

This tutorial is deprecated and being updated in the frame of Europlanet-2024 - see here 🙂

EPN2020-RI

EUROPLANET2020 Research Infrastructure

H2020-INFRAIA-2014-2015

Grant agreement no: 654208

Document: VESPA- WP6-2-011-TD-v2.0

Setting up an EPN-TAP service

Date: 2021-07-09

Start date of project: 01 September 2015 Duration: 48 Months

Responsible WP Leader:

Project co-funded by the *European Union's Horizon 2020 research and innovation programme*		
Dissemination level		
PU	Public	☐

PP	Restricted to other programme participants (including the Commission Service)	☑
RE	Restricted to a group specified by the consortium (including the Commission Services)	☐
CO	Confidential, only for members of the consortium (excluding the Commission Services)	☐

Project Number	654208
Project Title	EPN2020 - RI
Project Duration	48 months: 01 September 2015 – 30 August 2019

Document Number	VESPA-WP6-2-011-TD-v2.0 (65)
Delivery date	11 Mar 2016
Title of Document	Setting up an EPN-TAP service
Contributing Work package (s)	WP6
Dissemination level	PU
Author (s)	Stéphane Erard, Baptiste Cecconi, Pierre Le Sidaner, Florence Henry

Abstract: This tutorial shows how to install an EPN-TAP v2 service based on an intermediate table describing the data.

1.0

Document history (to be deleted before submission to Commission)				
Date	**Version**	**Editor**	**Change**	**Status**
18/11 /2013	Draft 0.1	Stéphane Erard		First Draft
	Drafts 0.2-0.4	several	many	(iterations)
14/1 /2014	Draft 0.5	Stéphane Erard	Improved case support Improved SQL support	Finalized for EPN-TAP v1 (imported from internal MS-Word document)
11 Feb 2016	0.6	Baptiste Cecconi	Improve formatting Adapted for EPNCore-v2 compliance	DRAFT
24/2 /2016	0.7	Stéphane Erard	Complete upgrade for v2, with walk-through example	DRAFT
31/3 /2016	0.8	Stéphane Erard, Nathanael Jourdane, Jean-Michel Glorian, Baptiste Cecconi	Many corrections + readability	DRAFT
9/4/2016	0.9	Stéphane Erard, Nathanael Jourdane, Jean-Michel Glorian, Baptiste Cecconi	Corrections during workshop. Now assumes DaCHS is installed in a Virtual Machine	DRAFT
5/6/2016	0.10	Stéphane Erard	Added lists of values and footprints	DRAFT
13/9 /2016	0.11	Stéphane Erard	Switch dev interface to public + minor fixes for delivery (end of year 1)	DRAFT
7/2017	1.0	Stéphane Erard	Outdated end 2017 - some sections still relevant though	DRAFT
4/2018	2.0	Stéphane Erard	Updated version, based on usage of EPN-TAP mixin (open 7/2017)	ISSUED

Table of Contents

Reference documents

- [RD1] EPN-TAP document, v1
  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
- [RD2] EPN-TAP v2 definition doc (in progress)
  EPNcore v2 (https://voparis-confluence.obspm.fr/display/VES/EPNcore+v2)
- [RD3] UCD + UType concept
  http://ivoa.net/Documents/cover/UCDlist-20070402.html
- [RD4] EPN-TAP installation for VESPA data providers / Tutorials:
  EPN-TAP Installation for VESPA Data Provider Tutorial
  Registering your VESPA EPN-TAP Server
- [RD5] DaCHS GAVO / German Astronomical Virtual Observatory. Contains the documentation and a tutorial for installation and configuration.
  http://vo.ari.uni-heidelberg.de/docs/DaCHS
- [RD6] IVOA Registry interface
  http://ivoa.net/Documents/RegistryInterface/
- [RD7] VirtisPDS IDL/GDL library:
  http://voparis-europlanet.obspm.fr/othertool.shtml
- [RD8] Units in the VO:
  http://wiki.ivoa.net/twiki/bin/view/IVOA/VOUnitsRFC or
  http://www.ivoa.net/documents/VOUnits/

Acronym list

EPNCore	Set of core parameters from EPN-DM, mandatory for EPN-TAP compatibility
EPN-TAP	Specific protocol to access Planetary Science data in Europlanet-VO, a variation on the TAP protocol
EPN-DM	Specific Data Model to describe Planetary Science data in Europlanet-VO
epn_core	Name of the table of a database which contains the EPN-TAP parameters. Required for EPN-TAP compatibility.
gavo	German Astronomical Virtual Observatory
DaCHS	GAVO Data Center Helper Suite (TAP server for your services)
IVOA	International Virtual Observatory Alliance
TAP	Table Access Protocol - general IVOA mechanism to access tabular data
ObsTAP	TAP protocol applied to the Observation Data Model of IVOA
ObsCore	Set of core parameters from the Observation Data Model of IVOA
ADQL	Astronomical Data Query Language, used to issue VO queries
PADC	Paris Astronomy Data Centre, at Paris Observatory - formely known as VO-Paris, hence the name used in the URLs
UCD	Unified Content Descriptor: defines measured physical quantities in the IVOA
utype	Description of data properties, in relation with a Data Model
VOSI	Virtual Observatory Support Interface

Introduction

This document provides practical guidelines to setup an EPN-TAP data service for non-specialists. It focuses on building the data service itself, not on software installation which is addressed in [RD4]. A simple example is used here to illustrate each step of the procedure: it will provide access to a series of individual files.

> **IKS info**
>
> The example service is based on spectral observations of comet 1P/Halley from the IKS instrument on board the Vega-1 spacecraft. The data were initially archived in the PDS as part of the International Halley Watch distributed on CD, then restored as an individual dataset in 2011. The data used in this service were retrieved from the PDS3 archive at PDS Small Bodies Node (2011 restored version) and updated with extra information recovered from the original team.
> The EPN-TAP service has been last updated in April 2018, and is described here: https://sites.lesia.obspm.fr/data-services/the-iksvega-1-dataset/

The procedure described here follows the recommendations from the DaCHS team. It involves the creation of a data table describing the service content, and the writing of a resource descriptor for DaCHS. Although this triggers the creation of a PostGreSQL database, no explicit SQL handling is required.

The database installed during this procedure can easily be used to feed a web site with a different interface, in particular through SPIP (procedure to be described).

DaCHS server

First of all, you need to use a VO framework which will handle the queries and answers to your service according to the TAP protocol. VESPA is supporting the use of DaCHS; although other VO frameworks (e. g., TAP library) support EPN-TAP services succesfully, those may require more work on your side.

We assume you have installed DaCHS on a Virtual Machine as described in EPN-TAP Installation for VESPA Data Provider Tutorial (see [RD4]). In this configuration, you will implement your service on your server in the guest machine, although you can access it from your usual system (host machine) - see Figure 1. In case of technical issues during the tutorial, some hints are provided in Annex I below.

? Pièce jointe inconnue

Figure 1

It is assumed that this DaCHS server is used as a "development server" - i. e. an area not accessible to external users, where you can test new functions. If you already have services open to users, it is better to have both a "production server" (open to your users) and an independent "development server" (not registered, and preferably not on-line) where you can freely test new design solutions without perturbing the existing service.

You will need to enter the gavoadmin password of your server at some point - you can retrieve it this way:

```
more /var/gavo/etc/feed
```

Note the value on the line:

password = xxxxxxxxxxxx

Example files

To get the reference example in working order, you can download the example files this way:

```
sudo apt-get install git
cd ~
git clone https://github.com/epn-vespa/DaCHS-for-VESPA.git
cd ~/DaCHS-for-VESPA/q.rd_examples/iks
```

This will clone the repository on your disk, including several examples of EPN-TAP services. The iks directory contains:

- dbiks2.pro - table building routine (IDL)
- indexiks.csv - basic data table in csv format (built by dbiks2.pro)
- catiksfiles.pro - specific PDS to VOtable conversion routine (IDL)
- IKS_mixin_q.rd - service resource descriptor

VO Tools

To retrieve and visualize the data you will use VO tools such as TOPCAT, Aladin, or CASSIS, depending on the type of your data. These applications run in the Java Virtual Machine. You need to download their installer or jar files (some clients may provide a standard application interface or a webstart link). You can test a jar file by clicking on its icon or by issuing a shell command from their directory:

```
java -jar <application>.jar
```

- Aladin (for images)
  http://aladin.u-strasbg.fr/java/nph-aladin.pl?frame=downloading
- TOPCAT (for tabular data)
  http://www.star.bris.ac.uk/~mbt/topcat/#install
- CASSIS (for spectra)
  http://cassis.irap.omp.eu/?page=installation

In this tutorial, we will also use the VESPA main query portal at Paris Observatory: http://vespa.obspm.fr

Data selection & service design

The very first step in building a service is to decide the content of this service: what are the data of interest, what is the proper organization of the data, is it consistent and correctly self-described?

In EPN-TAP, the data are presented in a table that describes data "granules" with several parameters, which are used to select data of interest for the user. "Granules" are the smallest data chunks described and accessible in the service; this granularity level must be identified from the start. In the common case where the data consist in a series of data files (e.g., telescopic images), granules usually are the individual files. When the data consist in series of scalar numbers describing objects (e.g., a list of orbital parameters), granules correspond to these objects and may be more difficult to identify (they can be targets, observations, etc… depending on the context).

IKS info

In the case of IKS, the dataset is a series of 101 calibrated infrared spectra of the comet (essentially the coma), plus 2 “reconstructed” spectra deriving from global analysis. All spectra are available as individual files. Most spectra use the same spectral range; one of the reconstructed spectra derives from the longer wavelength channel of the instrument and is included because it provides essentially the same kind of information (composition of the coma). In contrast, data from the imaging channel of the instrument are not included in this service, because they have a very different nature and scope: they provide intensity measurements of a modulated signal on a grid, and were mostly intended to derive a size estimate of the nucleus. For IKS, the main spectral data service will therefore only include the 103 infrared spectra, while the imaging channel data could be the scope of another service. The spectral data service will provide access to all individual spectra, which constitute the “granules” of the service.

• In EPN-TAP (v.2), all data products are described as distinct granules - they can be either individual data files (linked from the table) or sets of scalar parameters (included in the table). There is therefore at most one link to a file per granule. The only exception is the possibility to associate a granule to a small-size thumbnail, which is used to provide a quick-look in the search interface. "Previews" are more elaborate products which are treated as independent granules (e. g., reduced resolution images, a graphic representation of a table, etc).

• Some services contain related data products, e. g., a calibrated image and a map-projected version of the image, spectra and derived spectral parameters, tabular data and a graphic preview, etc. In such cases, data products derived from the same measurement are introduced as independent granules but are related through the same "observation". Instead, data products of similar nature (e.g., all calibrated spectra) belong to the same "group". All granules are therefore described separately, but may be associated in observations or groups (the grouping scheme is left to the provider's choice).

In practice, you have to define three ID parameters for each granule:

EPN-TAP parameter	Description
granule_uid	provides a unique ID for the granule in the service
obs_id	identifies granules related to the same initial measurement but containing different types of data (e.g.: raw and calibrated versions of the same file, measurement and associated geometry, etc)
granule_gid	identifies a group of products: it may be identical for all granules containing the same type of information for different observations (e.g., preview/native/calibrated/geometry/projection, etc). Usage is more flexible than for obs_id

Explicit strings are recommended in these fields, because they may be used as search criteria by your users; spaces are not allowed in these strings. Standard values of granule_gid are suggested in the table above.

Please remind that **granule_uid must be unique for the service to work:** otherwise the DaCHS server will issue fatal errors when ingesting the q.rd file later in the process.

IKS info

In the case of IKS, the service will provide access to an updated version of the PDS archive, which is completed with extra information and converted to a more handy format. In addition, it was decided to also maintain access to the historical PDS archive files. So there are two groups of products ("archived" for the PDS version and "corrected" for the updated version), and in all pairs the granules share the same observation ID (filename root, e.g. "iks030"). The unique ID is simply a combination of the two (e.g., "iks030A" and "iks030C").

An important part of service design is therefore to identify the groups of granules, and there is no unique way to do this. Group IDs are a handy way to separate different levels of data which are difficult to distinguish otherwise. A basic rule of thumb is to separate similar granules that you usually do not want to retrieve together as an answer to a request: several kinds of map projections, measurements vs associated geometry, different channel measurements, etc — especially when these files are not easily identified from other EPN-TAP parameters (e.g., images and spectra for instance are easily identified from the dataproduct_type parameter).

Gathering information on the data

The next step is to document the granules as accurately as possible, so that the user will be able to locate them from a query based on observational or operational parameters. The current tutorial describes a common situation where the information is first gathered in a table describing the content of the database, and saved in csv format (one line per granule / file). This data table will then be ingested to build the service table used with the EPN-TAP protocol. The initial data table can be written from IDL or a similar computing environment, or even manually; this is best done on your usual (host) machine to benefit from a familiar interface (keyboard layout, editor, etc). For situations where the table is already available in SQL, or should be retrieved from a collection of fits files, see the other tutorials.

IKS info

In the case of IKS, the quantities describing the data are spread among the file labels, the documentation of the PDS archive, the publication of the final results of the experiment, and other documents only available in the original team (LESIA, Observatoire de Paris).

Although an index file was included in the PDS dataset, it does not contain all the information included in the file headers. Since IDL is the only convenient environment to read data files provided in PDS3, we use an IDL routine to open all the files in sequence and parse the relevant information (the PDS files are read using the virtisPDS IDL library [RD7]). An extended index table is built from this, and is completed with extra information collected from other sources.

The IDL procedure `dbiks2.pro` reads all the IKS PDS data files, extracts the relevant information from the headers and stores it in a structure. References (instrument and mission names, etc) as well as extra information (exposure times, spectral resolution, phase angles) are incorporated manually. A URL pointing to the reformatted files is also included (see below). All the information required to build the service table is therefore available in this structure. The spectral parameters are still provided in native units, e.g., wavelengths in µm and time in UTC. However, complicated conversions would be best handled in the IDL routine.

The routine saves this structure in csv format, as `indexiks.csv`. This file is not self-documented (in particular units are not provided) and is only used to ingest data into DaCHS through the q.rd file, which can include part of the formatting/conversions.

The routine `dbiks2.pro` can be used as a template and adapted to similar cases, or automated further for more complicated cases.

Describing your data

The EPN-TAP protocol [RD1, RD2] is based on a set of mandatory parameters describing all databases uniformly. Although most parameters can be left empty when not relevant, they must be informed whenever possible to insure the accuracy of the search functions.

Remember that parameters with constant values can be initialized in the next step (q.rd file), as well as duplicate lines (e.g., for files provided in two different formats). This may simplify the current step.

Constraints on numerical parameters

Most numerical parameters have to be converted in standard units at some point, to make uniform queries possible on all databases. The space, time, and spectral coordinates are often of particular importance to identify data of interest inside a database - beware that they must honor specific conventions in the data description so that your service can be searchable. Those are listed in the EPN-TAP parameter documents (EPN-TAP V2.0 parameters).

- Spatial coordinates can be provided in various systems (Cartesian, Body-fixed, Celestial, etc). Each system follows specific conventions, e.g., longitude must follow the eastward convention in a body-fixed frame.
- Time is provided as UTC at the observer location, and formatted as Julian days.

- Spectral ranges must be provided as frequency in Hz in the service table.
- All floating-point parameters must be provided as double precision.

Constraints on string parameters

- Values of some parameters have to be selected from predefined lists (see EPN-TAP V2.0 parameters)
- In free format strings, the # character is not authorized
- Heading and trailing spaces are forbidden (they cause nasty errors in the interfaces, and must be checked). Spaces are best avoided in general.
- In names, acronyms, etc, upper cases are allowed when they follow official naming schemes (e.g., IAU target names); lower cases are required otherwise
- Multiple values are supported; lists are sequences of values separated by # (with no extra space)
- If you need to include "special" characters, UTF-8 encoding should do. Please check there are interpreted correctly at the end of the process.

IKS info

In the case of IKS, the time of observation was lost in early formatting of the dataset; it is provided here as the date of the encounter only, with no specific timing. In practice, the sequence of acquisition is described by the spacecraft-target distance (all observations were performed prior to closest approach). The spectral range was indicated in the PDS files, but no other information was available from the PDS archive. The spectral step is computed from the data. The actual spectral resolution and the exposure times are retrieved from the publication. The space coordinates are related to the observed area on the cometary nucleus, and are not known; these parameters are therefore left empty (this is not essential, since the nucleus is much smaller than the FoV even at shortest distance). Finally, the phase angles are interpolated for every observing distance from documents preserved in the original team.

The spacecraft-target distance is provided through the target_distance parameters (with min and max values identical) rather than as the third spatial coordinate, which is reserved to identify the observed area. This also avoids any ambiguity about the origin of the reference frame. Last, the target_time parameter is intended to cross-correlate simultaneous observations from another spacecraft or from the ground, if any. In this case it is left empty because the exact timing is uncertain.

• In addition to the granule identification parameters discussed above, some other EPN-TAP parameters must contain a value (see EPN-TAP V2.0 parameters) - those are mostly related to the service itself (service_title, dates, and dataproduct_type). Special care should be taken when selecting dataproduct_type, which is used essentially to identify the type of data (e.g., spectra from images) in a generic sense (a spectral plot is more a spectrum than an image). Its value must be selected from a predefined list.

EPN-TAP parameter	Example IKS value	Comments
service_title	iks	Schema name (see below), lower cases
creation_date	2013-11-17T10:41:00	As ISO time string. Value when the file is written (orginal PDS dates preserved)
modification_date	2013-11-17T10:41:00	As ISO time string. Date of latest modification
release_date	2013-11-17T10:41:00	As ISO time string. Value when the file is written (orginal PDS dates preserved)
dataproduct_type	sp	All data are spectra (from predefined list)

• Some EPN-TAP parameters describe the target, the origin of the data and the type of measurements, and must be informed when setting up the service. In general a standard value is required, from the reference sources indicated in the EPN-TAP documentation [RD1, RD2]. Those include:

EPN-TAP parameter	Example IKS value	Comments
target_name	1P	See below
target_class	comet	From predefined list
spatial_frame_type	body	From predefined list
instrument_name	IKS	As in NSSDC master catalogue (for space instruments)
instrument_host_name	Vega 1	As in NSSDC master catalogue (space missions) or IAU code (telescopes)
processing_level_id	3	CODMAC level

measurement_type	phot.radiance;em.wl	Introduces a UCD, which must be carefully identified from existing values [RD3] This one stands for spectral radiance, as a function of wavelength.

• Special care should be taken with target names: the names should follow the exact IAU spelling, including case, or be left in lower cases. This is particularly sensitive for exoplanets, comets, and asteroids (e. g., names like xP for periodic comets, main denomination for asteroids). Additional columns in the table can be used to store alternative target names/spelling.

• Some parameters must take values selected from a predifined list, e. g., dataproduct_type is encoded according to the EPN-TAP protocol definition [RD2].

• Specific parameters can also be included in addition to the standard EPN-TAP parameters whenever relevant. They will be available to search inside an individual service, but they obviously cannot be used to perform searches across several services. If this is needed, first check if something similar exists among the optional parameters; using the same optional parameters will ensure consistent queries among related services.

• It is important to realize that the TAP mechanism can only query the service table, not its header. All relevant parameters must therefore be present in the service table with proper units.

IKS info

In the case of IKS, specific parameters include an acquisition number related to spacecraft operations, which is maintained for completeness. Optional parameters Sun and Earth distances are also included for in the table, although there are constant throughout the dataset (within the limits of the available accuracy) - they couldn't be accessed via the TAP mechanism otherwise.

Finally, a more explicit target name (Halley) is provided through the optional parameter alt_target_name.

Linking data files

An important parameter is the **access_url** which points to the files. When present, it must be associated with other parameters describing the file: **access_format** and **access_estsize**. However, this set of parameters is optional, and can be replaced by the data themselves in the case of a small data table of scalar values - allowing to perform searches on the data themselves.

IKS info

In the case of IKS, the access_url parameter provides the location of the data files: either at PADC (for the updated, reformatted files; see below) or at the Small Bodies Node of PDS (for the original PDS versions). The corresponding access_format values are votable and ascii - the latter because the URL points to the ascii data area of a PDS3 file (with no label).

Note that the access_url parameter can point to a script, rather than a file. This may be a handy way to support exotic or non-standard formats, and to perform conversions on the fly. For instance, scripts are used in several services to convert csv or ascii files to VOtable. Scripts may also be used to extract the data of interest from another database, from larger files, or to send queries on servers via specific protocols (e.g., WMS, das2stream, etc). The access_format parameter always refers to the returned product, and is used to pick up a visualization tool for the data.

Providing footprints

Footprints provide significant added value to services distributing spatially extended data (images or spectral cubes), either on the sky or on planetary surfaces. The C1/C2 parameters provide footprints as a simple bounding boxe (e.g., min/max longitudes and latitudes) for quick searches. However, this system produces many false alarms and is not adequate for accurate searches.

In addition, EPN-TAP uses the s_region parameter to provide 2D footprints according to pgsphere syntax. Such footprints can be used for spatial searches and can be displayed in plotting tools (e.g., Aladin). To define footprints, you first have to sample the contour of your data products - at least the corners are required for a polygon, but more points along the edges would provide more accurate results. Points must be listed in sequence and in direct / anticlockwise sense. If your data are on the sky, use RA and Dec values in degrees; on a planetary body, use eastward longitudes. If your footprints are not polygons (points, circles…), other geometric objects may be available (see the pgsphere documentation).

You then enter the list of points in the parameter s_region of every granule with syntax: '{(lon1,lat1), (lon2,lat2), … }' (with no quotes). Arguments lon and lat must be provided as: 10.d, 5d, etc, where character d stands for degrees.

The VESPA user interface will allow you to look for inclusions and intersections with a search box (use the Direct Query mode of VESPA). Comparison between footprints of various data products will be also be available in the future.

The s_region parameter is declared with type 'spoly' when creating the table. In case you need to manipulate the table in a later stage, avoid any function involving comparisons of s_region parameters (e.g., use UNION ALL instead of UNION, avoid DISTINCT, etc).

Building the resource descriptor

Once the data are selected and formally described in a csv table, you are ready to set up your service.

The service will be installed in a database "schema". The schema name is that of the service. The EPN-TAP protocol expects a service table called <schema>.epn_core (see Figure 2) - this table will be queried by the user interfaces (TAP clients), and has to be formatted according to the EPNCore standard.

DaCHS uses a "resource descriptor" to set up services, an xml file always called q.rd. This file calls a "mixin", which provides the complete definition of standard EPN-TAP parameters, and uses a "grammar" to handle data ingestion. Some python code can also be included to modify the original data table, e. g., to convert units or to duplicate rows when variations of the same granules are provided in the service. Upon import, the q.rd file will create the database under postgreSQL, the service table, and the service itself. Again, this file is best written in your usual (host) machine, then transferred to the virtual (guest) machine.

Beware that the **table and schema names are case sensitive in SQL**. A good practice is to always provide these strings in lower case — otherwise, DaCHS may block at the end of the process.
“Special” (accentuated / diacritic) characters must be avoided in this file, even in descriptions (they will be displayed by various tools, which often expect ascii formatting).

Figure 2

In the present context, the q.rd file is made of 3 parts: general metadata; table structure; ingestion rules. You can use the IKS_mixin_q.rd example file as a template and adapt it to your needs.

General metadata

These fields are intended to provide reference to the service and describe it in the registry (longer description here: Building the resource descriptor for your EPN-TAP service in DaCHS)

The first line provides the name of the schema, and must be replaced by that of your service (case sensitive).
The description of the service that follows must be adapted — lines starting with <meta name =… >. Only text is allowed here (no macro, e. g. \metaString). Use a generic service for contact.email. Some of these elements are only allowed to appear in a single instance (e. g. contact.name, contributor.name) - include a list of values if required.
The utype element provides reference to the protoocol in use.
Subject values are best taken from the Unified Astronomy Thesaurus. They must include at least one of the following top-level values for Solar System services: Solar system astronomy, Solar physics, Exoplanet astronomy (see this page for all possible values and subdivisions).

```
<resource schema="**iks**">
  <meta name="title">**Full title**</meta>
  <meta name="description" format="plain">**Full description (~ 5 lines), possibly with bib
reference.**</meta>
  <meta name="creationDate">2017-12-11T19:42:00Z</meta>
  <meta name="subject">comet</meta>
  <meta name="subject">spectroscopy</meta>
  <meta name="subject">1P Halley</meta>
  <meta name="subject">Solar system astronomy</meta>
    <meta name="creator.name">**Your name**</meta>
    <meta name="contact.name">**Somebody responsive**</meta>
    <meta name="contact.email">**mail of a generic service**</meta>
    <meta name="contact.address">**Postal address**</meta>
    <meta name="instrument">**Instrument name/acronym**</meta>
    <meta name="facility">**Instrument host name**</meta>
  <meta name="source">**bibcode of main reference**</meta>
  <meta name="contentLevel">General</meta>
  <meta name="contentLevel">University</meta>
  <meta name="contentLevel">Research</meta>
  <meta name="utype">ivo://vopdc.obspm/std/epncore#schema-2.0</meta>

    <meta name="coverage">
        <meta name="waveband">Infrared</meta>
        <meta name="Profile">none</meta>
    </meta>
```

The element referenceURL is used to link to an existing web site describing the service (specify that of your institute if no such web site exists).

Table structure

The table element declares the service table name, then introduces the epntap2 mixin that provides a complete definition of all the mandatory EPN-TAP parameters. Optional parameters in use need to be declared in the second line, again with definition provided by the mixin (only declare those used in your service). Finally, non-standard parameters must be declared with all their metadata in successive blocks, as shown in the example (acquisition_id).
• The mention of spatial_frame_type="body" is a default that applies even when no spatial_frame_type is applicable - The value "body" must be replaced by the relevant value otherwise.
• When defining non-standard parameters: all floating point parameters must be double precision; units must conform to the “Units in the VO” document [RD8].

```
     <table id="epn_core" onDisk="true" adql="True">
         <mixin spatial_frame_type="body"
         optional_columns= "access_url access_format access_estsize access_md5
alt_target_name publisher bib_reference" >//epntap2#table-2_0</mixin>

         <column name="acquisition_id" type="text"
             tablehead="Acquisition_id"
               description="Extra: ID of the data file in the original archive"
               ucd="meta.id"
             verbLevel="2"/>
     </table>
```

Ingestion rules

The "data" element of the q.rd provides:

• A reference to the data source, in this case the path to the csv data file;
• The csvGrammar element contains the global ingestion rules:
  • A first rowfilter element providing the name of the service table and a global assignement of the csv file columns to EPN-TAP parameters (when the csv columns have parameter names);
  • A second rowfilter element handling special conditions may appear here. This contains a python code executed upon ingestion.
• The table element contains a rowmaker element providing extra ingestion rules:
  • <var> elements define intermediate variables for future use.
  • Mandatory parameters associated to table columns bearing a different name, or modified from the data table, must be associated using <bind> elements. Modifications can be introduced in the bind element (python style) or in the rowfilter code.
  • Optional and non-standard parameters must be defined in the <var> element of rowfilter code in general. Depending on the epntap2 mixin version in use, they may require to be defined in the <bind> element (check messages upon import).

```
        <data id="import">
        <sources>data/indexiks.csv</sources>
        <csvGrammar>
            <rowfilter procDef="//products#define">
                <bind name="table">"\schema.epn_core"</bind>
            </rowfilter>

            <rowfilter name="addExtraProducts">
                <!-- Duplicate input granules to provide 2 alternative versions -->
                <code>
                        # whatever processing, see below
                </code>
            </rowfilter>
        </csvGrammar>

        <make table="epn_core">
            <rowmaker idmaps="*">
                <var key="alt_target_name" source="target"/>
                <var key="bib_reference" source="ref"/>

                <apply procDef="//epntap2#populate-2_0" name="fillepn">
                    <bind key="time_min">dateTimeToJdn(parseISODT(@time_obs))</bind>
                    <bind key="time_max">dateTimeToJdn(parseISODT(@time_obs))</bind>
                    <bind key="time_scale">"UTC"</bind>
                    <bind key="spectral_resolution_min">2.99792458E14 *(float(@sp_res_max)
+float(@sp_res_min))/ 2 / float(@sp_max)**2</bind>
                    <bind key="spectral_resolution_max">2.99792458E14 *(float(@sp_res_max)
+float(@sp_res_min))/ 2 / float(@sp_min)**2</bind>
                    <bind key="spectral_range_min">2.99792458E14 /float(@sp_max)</bind>
                    <bind key="spectral_range_max">2.99792458E14 /float(@sp_min)</bind>
                    <bind key="time_min">dateTimeToJdn(parseISODT(@time_obs))</bind>
                    <bind key="time_max">dateTimeToJdn(parseISODT(@time_obs))</bind>
                </apply>
            </rowmaker>
        </make>
    </data>
```

IKS info

In the case of IKS, all required conversions from the data table are performed in the bind elements: wavelengths in µm to frequencies in Hz, and dates from ISO string to Julian Days. The conversion coefficients must be used as in the example, because the VESPA portal will perform the exact inverse conversion, based on accurate physical values. This example can be used as a template for other databases.

The python code in the rowfilter element allows for post processing of the input table. In the case of IKS, the two blocks duplicate the granule entry and adjust the parameters for the two versions of the spectra (VOtable and PDS formats). Parameters are referred to with the @ prefix. Processing can also be triggered by conditions on the input parameters, see DaCHS doc or other examples.

```
                <code>
                        # VOtable version
                        @granule_uid = @rootname+"C"
                        @granule_gid = "corrected"
                        @access_format = "application/x-votable+xml"
                        @access_estsize = "19"
                        @access_url = @a_url
                        @file_name = @rootname+".xml"
                        @thumbnail_url = @a_url+".png"
                        @creation_date="2013-11-17T10:41:00.00"
                        @modification_date="2016-04-28T15:43:00.00"
                        @release_date="2013-11-17T10:41:00.00"
                        yield row.copy()

                        # native version
                        @granule_uid = @rootname+"A"
                        @granule_gid = "archived"
                        @access_format = "text/plain"
                        @access_estsize = "4"
                        @access_url = @o_url
                        @file_name = @rootname+".tab"
                        @thumbnail_url = ""
                        @creation_date="1993-11-10T07:54:00.00"
                        @modification_date="1993-11-10T07:54:00.00"
                        @release_date="1993-11-10T07:54:00.00"
                        yield row
                </code>
```

Beware that some conversions to epn_core standard are trickier than they seem. In particular, conversions from wavelength to frequency, or from calendar time to JD, are best copied and adapted from the IKS example to minimize errors and save time. In any case, check your results carefully to ensure proper access to your data.

Converting the data files (optional)

In many instances, the original data files are not available in a format easily handled in the VO environment. The usual VO formats are:

- FITS images and tables are OK if their headers are sufficiently informative.
- VOTables are xml files handled by all VO applications, and are particularly useful for tables, spectra, and catalogues.
- Tables can be stored as ascii or CSV files but may require specific extensions (.asc or .csv in general). In practice, these files are not self-described and are therefore difficult to use in the VO. Besides, files downloaded individually will contain no mention of origin, therefore this is not a preferred choice.
- CDF files are supported only in TOPCAT.
- Standard image formats such as PNG, JPEG, etc are usually supported by VO tools. However, compressed image formats are in general acceptable only for low accuracy previews and thumbnails, not for a science usage.

In any case, an important matter is that data files must be entirely self-documented, e. g., they must contain (in addition to orign and credits) all metadata providing important observation conditions (a subset of those contained in the table) — otherwise files downloaded individually would be hardly usable. This is routinely done using the PARAM elements in VOtable headers, or standard keywords in fits headers; relying only on the file name is a bad idea.

For this reason, it may be convenient to convert the data files to VOTable or fits format. This will not only permit self-documentation of the data files, but will also make them loadable directly by the VO tools; your service will then fully benefit from the VO interface of VESPA. As noted above, conversion can also be performed on the fly by using the access_url parameter to point to a script.

If you have to convert your files, notice that there is no need to convert the units used in the data files to the EPNCore standards - this is only required in the epn_core table to support queries.

IKS info

The original IKS archive from the PDS is stored in PDS3 format with detached labels.

The IDL routine catiksfiles.pro reads all the PDS3 data files present in a directory and convert them to VOTable. The files are read using the virtisPDS IDL library [RD7]. Writing the xml files is done using the IDL object IDLffXMLDOMDocument. Descriptive information is taken from the PDS labels, and is completed with extra information (in the case of IKS, target distance and obs ID are included, while phase angle, instrument, instrument host name, resolution, etc could be added). The data area of the VOTable is stored in ascii to preserve relatively easy reading. Again this routine can be used as a template for other databases (other solutions to write VOTables are available in IDL).

When converting the data files to VOTable, special care must be taken to describe the data area: all quantities are associated with a name, unit, datatype, and UCD. UCDs must follow the VOTable/IVOA standard [RD3]. A tool to help selecting adequate UCDs is available here:

http://dc.zah.uni-heidelberg.de/ucds/ui/ui/form

IKS info

When restoring an older dataset, it is a good idea to check the consistency of the data. In the case of IKS, an inconsistent scaling factor in radiance was spotted by comparison with the published results, and corrected in the service (radiance of order 10^{-6} mentioned in the PDS labels, while the paper clearly shows a factor of 10^{-7} except for the long wavelengths spectrum).

Installing your service

If you've edited the files in your host machine, you first need to copy them on the guest in the correct location. You first have to create a directory in the gavo tree, with the exact name of the schema. Respect the path given below (/var/gavo/inputs/iks/q.rd, where iks is replaced by your service schema name) and don't change directory names or move files after after a first import. Yo u may need to adjust autorizations.

```
On the host side:
scp -P 2222 q.rd  toto@localhost:/var/gavo/inputs/iks/q.rd
scp -P 2222 indexiks.csv  toto@localhost:/var/gavo/inputs/iks/data/.
```

Once your q.rd file is ready and properly located check its validity using the command:

```
On the guest side:
cd /var/gavo/inputs/iks
sudo gavo val q.rd
```

If the syntax is correct, this will answer:

```
q.rd -- OK
```

If you need to edit the files, do it in your guest machine and scp again to the host.

Then import the file:

```
sudo gavo imp q.rd
```

If import is successful, the output will indicate:

```
Making data import
Starting /var/gavo/inputs/iks2/data/indexiks.csv
Done /var/gavo/inputs/iks2/data/indexiks.csv, read 206
Shipped 206/206
Rows affected: 20
```

Any other answer is suspect. Note that the sudo command is mandatory here - error messages may be hidden otherwise.

If you get a persistent message such as
`UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 91: ordinal not in range(128)`
try deleting the table, ingest it again, then import the q.rd again and restart the server.
If it is still there, try doing this several times.

When import is done, restart the DaCHS server:

```
sudo gavo serve restart
```

You can now test your service in private mode. Make a test query through your web browser using the following URL (replace serveraddress and myschema by the correct values; in test mode, serveraddress should be 127.0.0.1:8000):

http://serveradress/__system__/tap/run/tap/sync?REQUEST=doQuery&LANG=ADQL&QUERY=SELECT * from myschema.epn_core&FORMAT=votable/td

If you need to reimport the q.rd file, always edit or replace the previous version in the directory where it was first located, and import it again - never change the location of this file or the directory name. If you need to move (or remove) a service, go to the adequate directory and type:

gavo drop -f q.rd

If you moved the service, you may need to restore the initial conditions. In case of big trouble, try this:

gavo purge myschema.epn_core
sudo gavo imp q.rd

Testing / querying your service

Checking your service output

The first test is to check your service output using the validator at PADC:

http://voparis-validator.obspm.fr/

- Select "EPN-TAP2" in the "Specification" menu, click LOAD
- Enter your service schema name in the corresponding field (e. g., iks.epn_core) - please check and remove any extra space in this field
- Provide the URL of your test server in the empty field “URL to validate” at the bottom, e.g. http://yourserver.wherever/__system__/tap/run/tap
- click VALIDATE

Beware that messages are relative to the output VOtable from the server. This VOTable is formatted based on the definition of the epn_core table as described in your q.rd file. This needs to be fixed in case of problem, but messages from the validator do not directly apply to this table. In case of problem, go back to the above definitions and check every step - in particular, check the extra parameters in your service, not predefined in the epntap2 mixin (beware however that in some cases the validator may be out of synch with the epntap2 mixin, so that "fatal errors" are actually benign).

Tests using various clients

A client is a piece of software allowing the user to write queries, and handling answers. Among possible clients, the VESPA portal is a web form optimized for EPN-TAP, available here: http://vespa.obspm.fr

The VESPA portal can access services which are not declared, provided they are on the internet with a fixed address.
The TOPCAT tool also includes a generic TAP client which may be used for testing, although the interface is less user-friendly.
When testing, check that all parameters appear as you wish with the proper units, and that "special" characters (preferably entered with UTF-8 encoding) display correctly in the interfaces.

Illustrated explanations are available on this page:

https://voparis-confluence.obspm.fr/display/VES/Checking+your+VESPA+service

Tests using DaCHS

The DaCHS server includes a low-level client which can access its own services (therefore, you don't have to specify a server address). It accepts the same ADQL requests as above.

```
On your Virtual Machine, the DaCHS query form is located here:
http://127.0.0.1:8000/__system__/adql/query/form
```

Tests using TOPCAT

You can easily test the service access with TOPCAT.

A graphical tutorial about using TOPCAT to access EPN-TAP services is available here: http://voparis-europlanet.obspm.fr/utilities/Tuto_TopCat.pdf

- open TOPCAT

- chose menu VO->Table Access Protocol (TAP) Query
- write the url of your server in the field “TAP URL” at the bottom of the dialogue

To access the DaCHS server installed on a Virtual Machine in EPN-TAP Installation for VESPA Data Provider Tutorial (test mode), enter:

http://127.0.0.1:8000/__system__/tap/run/tap

then click “Use Service”.

TOPCAT will use the specifications of TAP to browse the server and will display all the tables available for query.

Select “iks.epn_core” from the list. All the fields from this table will display under Columns in the right panel.
You can retrieve all granules in TOPCAT by typing the query in the ADQL text field at the bottom:

```
select * from iks.epn_core
```

Type "Run Query", the table will load in the Table list.
Click on the 4th icon from the left in TOPCAT (“Display table cell data”). The complete service table will display in TOPCAT.
Go to column access_url, click 3 times on a cell to select and copy it. Select File->Load Table. Paste into location and click OK.
The result file (a spectrum) is loaded into TOPCAT.
Select the icon “Scatter Plot”, a graph will plot. If multiple axes are present, you can easily select the axes to be plotted. The rainbow icon allows you to display a third column using colors.

Tests using the VESPA portal

Your service is not yet registered, but if it is installed on a machine directly connected to the internet (or during the implementation workshops) you can test it from the VESPA portal. Alternetively, you can install the VESPA client on your VM (see Installing a local VESPA client).

To retrieve your IP number on the private network during implementation workshops, connect on the vespa wifi and type ifconfig in terminal (say: 10.42.0.1)
Otherwise, use the IP number of the machine where you server is located.

Click the button “Custom resource” on the top of the VESPA page then fill the “Resource URL” and “Schema name” fields with the proper indications. If the test IKS service is installed on your Virtual Machine, this is (using the IP number retrieved in the previous step):

```
Resource URL = http://10.42.0.1:8000/tap
Schema = iks
```

You can now issue a test query related to your service in VESPA, using standard EPN-TAP parameters: time between 1986-03-05 and 1986-03-12, target type = comet, dataproduct type = sp (spectrum). VESPA will transform your entries into an ADQL query:

```
SELECT * FROM ... WHERE (dataproduct_type LIKE '%sp%') AND (target_class LIKE '%comet%')
AND time_max <= 2446501.50000000 AND instrument_name = 'IKS' AND time_min >=
2446494.50000000
```

This query is then embedded into an EPN-TAP request sent to your server.

The “Query results” page describes the service in test and provides the number of results. If matches are found, the box is displayed in green (grey if no result, or red if the service is in error). Click on the “Display results” tag to access a list of individual results.

The “Service results” page will display the main parameters of the results, including access_url if the data are linked from the epn_core table. Other parameters (including non-mandatory EPN-TAP parameters) can be displayed using the “Show all” button. From there, your data can be selected and sent to VO tools (see the Help button on this page for help on the interface capabilities).

From the Query results page (after a first query) you can click the “Advanced query form” tag in the service box to access non-mandatory parameters present in your service. This will display a new query form completed with extra parameters present in the service. URLs and filenames can also be queried this way.

Registering a service - to be reviewed

To make your service publicly available you must declare it in a registry, which is queried by all VO clients. The declaration in the IVOA and EPN registries is made though an xml file containing a VOResource xml extension [RD6], or through a web interface. Two main registries exists, which propose an on-line service declaration tool:

NVO: http://nvo.ncsa.uiuc.edu

ESAC: http://registry.euro-vo.org

Alternatively, some institutes provide “publishing registries” where local resources are declared; these registries are regularly harvested by the main ones, once they are declared themselves. The DaCHS framework in particular allows institutes to set up such a solution, which is available e. g. at PADC for local services. In any case, it is important to check the services using on-line validators, before making them public.

IKS info

The IKS data service uses a basic declaration form that provides the information required for most small services, and can be adapted to your needs.
The file is called: vopdc_obspm-lesia-epn-iks.xml

This template contains several parts:

- An element providing references to all xml definition files involved (<ri:Resource…).
- Several identification elements (title, short name, identifier)
- A curation bloc
- A content bloc
- A capability bloc

Each resource has an ivo-id (for more details, read the dedicated service identifier page). A typical ivo-id for your service is:
ivo://my_institute/my_laboratory/my_team/myservice/epn

Two specific constraints related to EPN-TAP must be respected:

1- First the service name must be of the form myservice/epn where *myservice* is the schema name of the database (e. g., “iks /epn”). The complete service ID will be:

ivo://my_institute/my_laboratory/my_team/myservice/epn

2- Second, the “short name” tag in the xml file must be identical to the schema name used by the service (case sensitive).

No explicit error will occur in DaCHS if those two conditions are not met, but the service may not function correctly.

• Credits are given in the Curation block:

- The “Publisher” of the service is the institute or organism providing access to the service, i. e., a data center. It is identified using the same ID as above (ivo-id) plus a human readable string (value of the tag). For IKS this is VO-Paris Data Center.

- Only one “Creator” tag is used in EPN-TAP, and introduces the original data provider (responsible for data acquisition, experimental setup, etc). For a space experiment, this would normally be the PI team.

- The “contributor” tags may be used to acknowledge other contributions, most notably the institute or individual who actually prepared the service. IKS is a special case of restored archive, therefore several contributions are listed: LESIA was in charge of the instrument, operations and data calibration, IKI (in charge of the spacecraft) provided a first version of the archive, and the data were retrieved from an intermediate preprocessed archive at the PDS Small Bodies Node. The individual who actually set up the service is also mentioned as contributor.

- The “Contact” tag refers to an individual or a group supporting the service, depending on local policy. In any case, the email address provided here is normally a generic helpdesk.

Variations on these rules are allowed in the registry declaration file, but may not reflect properly in the VESPA interface.

• The Content block provides general information, which is printed in the service boxes after a query from VESPA.

- The “Subject” tags are keywords extracted from the IVOA thesaurus

- The “Description” provided here is the one printed in VESPA’s description of the service.

- The “Source” tag introduces a bibliographic reference in the form of a BibCode (typically from ADS)

- The “ReferenceURL” tag provides a link to a web page describing the service - it must be informed, so you have to put here the web page of your institute if no dedicated web page is available.

- The “RelatedResource” tag provides reference to publishers of related services.

• In the Capability bloc, the “AccessURL” tag provides the URL of the TAP server to be queried (related to your installation of DaCHS/gavo).

Many other possibilities are supported by the registry interface. For instance, it is possible to restrain the access to parts of a service if some data are still in proprietary period. Again, we refer to the DaCHS tutorial to handle such actions:

http://vo.ari.uni-heidelberg.de/docs/DaCHS/tutorial.html

Registering to VESPA

The above procedure publishes your service in the IVOA registry, which makes it accessible from TOPCAT and other standard tools.

However, you also need to drop a mail to the VESPA team to make your service accessible from the VESPA portal with other validated services. The VESPA team will perform a manual review of the service and check consistency with other related services before publication.

Once the registry declaration is validated your service will seen by the VESPA, and coordinated queries will be possible. Use the default button “All VO” on top of the VESPA portal first page. The result will be a list of available services. Those containing answers to your query will be displayed in green and you will be able to select results from individual services.

Annex I : possible issues

• When copy/pasting the above examples, beware of special spaces! In case of problem, erase and replace them manually.

• Also remove any trailing and heading space in strings (especially in granule IDs), they can block the interfaces.

• The version of postgresql used in examples is 9.4. You may have to adapt some commands in the above examples to the version number in use.

• Some installations of java on OS X / Mac are known to be unstable - in particular v1.7 produces numerous graphic bugs and incompatibilities. TOPCAT used with the jdbc extension is particularly sensitive to this, as well as some other VO tools. Both versions 1.6 and 1.8 appear to work smoothly. If the interface looks unstable, keep the window size unchanged to minimize graphic mismatches.

• jdbc version must be consistent with your versions of both postgresql and java, see here: https://jdbc.postgresql.org/download.html.

It can be retrieved e.g. (check for the correct version) with: wget https://jdbc.postgresql.org/download/postgresql-9.4.1208.jre6.jar

• If the jdbc is ran from inside a Virtual Machine, you first need to open an ssh tunnel in order to reach the distant database from the host:

ssh <user>@127.0.0.1 -p 2222 -L 5432:127.0.0.1:5432

(see EPN-TAP Installation for VESPA Data Provider Tutorial — Part 1)

• Under Windows + Cygwin, lauching the jdbc requires a different syntax:

java -cp "`topcat-full.jar;postgresql-9.1-903.jdbc4.jar" -Djdbc.drivers=org.postgresql.Driver uk.ac.starlink.topcat.Driver`

• To log on your VM from the guest terminal, type

ssh -p 2222 <user>@localhost

• To copy files from your host machine to the guest, use:

scp -P 2222 *.sql <user>@localhost:~/iks/.

(note this is **uppercase P** — this is different from the ssh command)

• A possibly simpler solution to produce VOTables under IDL/GDL is to use the write_vot.pro routine, which is a wrapper routine of the stilts/TOPCAT library: https://github.com/epn-vespa/IDL_VOtable.

Other solutions involve scripts in python, etc.

• If you get this error: '*ascii' codec can't decode byte 0xc3 in position 19: ordinal not in range(128)* or similar, you can try to re-import the q.rd file:

```
sudo gavo val q.rd
sudo gavo imp q.rd
sudo gavo serve restart
```