Setting-up an EPN-TAP service: Tutorial for Beginners

Introduction

Setting-up an EPN-TAP service means publishing a table in a server (DACHS for this tutorial) describing data we want to distribute. This table references each piece of information with a maximum amount of attributes by filling parameters defined by the standards of EPN-TAP called epn_core view. The EPN-TAP standards have mandatory parameters (most of them could be left empty) and optional parameters, finally additional parameters could be defined to provide specific information. EPN-TAP is a way to interrogate registered services published in different DaCHS servers using parameters or ADQL queries.

This tutorial will provide a very simple example to help beginners in understanding how they can set up an EPN-TAP service. In this purpose, we will set up a service called "Planets" providing information on the solar system planets. It will only document the method using a CSV (Comma Separated Values) file to fill the table.

For this example, we want to display these data on our service :

name	mean radius (km)	mean radius uncertainty (km)	equatorial radius (km)	equatorial radius uncertainty (km)	polar radius (km)	polar radius uncertainty (km)	rms deviation (km)	elevation max (km)	elevation min (km)	mass (kg)	distance to primary (km)	
Mercury	2439.7	1.0	2439.7	1.0	2439.7	1.0	1	4.6	2.5	3.3014 E23	57909227.	1
Venus	6051.8	1.0	6051.8	1.0	6051.8	1.0	1	11	2	4.86732 E24	108209475.	-5
Earth	6371.00	0.01	6378.14	0.01	6356.75	0.01	3.57	8.85	11.52	5.97219 E24	149598262.	2
Mars	3389.5	0.2	3396.19	0.1	3376.	0.1	3.0	22.64	7.55	6.41693 E23	227943824.	2
Jupiter	69911.	6	71492.	4	66854.	10	62.1	31	102	1.89813 E27	778340821.	9.
Saturn	58232.	6	60268.	4	54364.	10	102.9	8	205	5.68319 E26	1426666422.	1
Uranus	25362.	7	25559.	4	24973.	20	16.8	28	0	8.68103 E25	2870658186.	-1
Neptune	24622.	19	24764.	15	24341.	30	8	14	0	1.0241 E26	4498396441.	1

Steps to follow

First, a virtual machine hosting a DaCHS server must be set up. For simple services like this one, metadata can be referenced in a CSV file containing information to include, it is also possible to use a Python script to generate metadata. Then, a Resource Descriptor (RD) must be written in XML. This RD aims to define columns, read the CSV file and fill a table on DaCHS server. Finally, the service must be registered on the IVOA registry which makes the link between EPN-TAP services and VESPA query interface.

• Install the server on a virtual machine

First, it is necessary to have settled-up an EPN-TAP server. The method to set up a virtual machine hosting a DaCHS server is described in this tutorial: EPN-TAP Server Installation for VESPA Data Provider Tutorial . You can also build it on a Docker container.

• Define granules

To publish your own service, you have to define granules. Granules correspond to table rows, they represent the smallest piece of information accessible in the service. A granule can be one file (linked with the optional parameter access_url) or a set of parameters described into the table. A granule must have one unique identifier which is a primary key for the table (the mandatory parameter for this identifier is granule_uid). Each different type of granule must have an identifier (granule_gid parameter) so it is possible to group data by type.

In our example, each granule has the same type and corresponds to one planet. So, basically the granule_gid and granule_uid will be respectively "Planet" and the name of the planet. There is no need to define an access_url because for this example, there is no file linked, each granule corresponds to a set of parameters.

- ## Define the epn_core parameters

Once granules are defined, you have to browse the EPN-TAP V2 parameters and their description to see what could be informed. In order to clearly define organization of the final table, it is advised to fill a scheme with the EPN-TAP parameters on one side and what you decide to put into each on the other side (available here in xls format). A large part of the mandatory parameters could be left empty, other parameters could be defined as additional-columns.

- ## Create a CSV file containing metadata

Metadatas which are varying from one granule to the other or cannot be post-treated in the resource descriptor must be referenced in a CSV file (if this is the method chosen). The first row has to be the column's names, then each row will set the values for a granule. This CSV file will further be read by the RD to fill the table in DaCHS. You can use your favourite programming language to create this CSV file.

For this example, we give the link to a hand-written CSV, you can download it and the associated Resource Descriptor from github:

> *https://voparis-gitlab.obspm.fr/workshop-2021-material/planets*

- ## Building a resource descriptor

The Resource Descriptor (RD) is an XML-written file named q.rd which gives DaCHS the way to fill the table from the CSV file, it follows DACHS Standards for EPN-TAP. Other information on the RD building are available here.

Structure of the Resource Descriptor

The RD file, named q.rd has this structure :

```
<resource schema="...">
   <meta .../>
       ...
   <meta .../>

   <table ...>
      <mixin .../>

      <column .../>
          ...
      <column .../>
   </table>

   <data id="import">
      <sources .../>

      <csvGrammar> <rowfilter procDef="//products#define"> <bind name="table">"
\schema.epn_core"</bind> </rowfilter> </csvGrammar>

      <make table="epn_core">
         <rowmaker idmaps="*">
            <map key="...">...</map>
               ...
            <map key="...">...</map>

         </rowmaker>
      </make>
   </data>
</resource>
```

The tag **<resource>** encompasses the others. First, **<meta>** data are filled, then, **<table>** is defined containing <mixin> reference and <column> elements defining extra-columns. Into the tag **<data>**, data ingestion rules are set, the path of the sourcefile is defined into <sources> and the <csv grammar> is specified. The **<make><rowmaker>** content describes how the mandatory and added columns will be filled. **<map>** attributes set columns values and fill the table.

Meta tags

The first part is a set of meta tags with different attributes which defines global characteristics of the table. Meta tags aim to describe the service in the registry.

```
<resource schema="planets">
    <meta name="title">Characteristics of Planets (demo)</meta>
    <meta name="description" format="plain">
    Main characteristics of planets. Data are included in the table, therefore most
relevant parameters are non-standard in EPN-TAP. Data are retrieved from Archinal et
al 2009 (IAU report, 2011CeMDA.109..101A) [radii] and Cox et al 2000 (Allen's
astrophysical quantities, 2000asqu.book.....C) [masses, heliocentric distances, and
rotation periods]. </meta>
    <meta name="creationDate">2015-08-16T09:42:00Z</meta>
    <meta name="subject">solar-system-astronomy</meta>
    <meta name="subject">planetary-science</meta>
    <meta name="subject">solar-system-planets</meta>
    <meta name="subject">periodic-orbit</meta>
    <meta name="copyright">LESIA-Obs Paris</meta>
    <meta name="creator.name">Stephane Erard</meta>
    <meta name="publisher">Paris Astronomical Data Centre - LESIA</meta>
    <meta name="contact.name">Stephane Erard</meta>
    <meta name="contact.email">vo.paris@obspm.fr</meta>
    <meta name="contact.address">Observatoire de Paris VOPDC, bat. Perrault, 77 av.
Denfert Rochereau, 75014 Paris, FRANCE</meta>
    <meta name="source">2000asqu.book.....C</meta>
    <meta name="contentLevel">General</meta>
    <meta name="contentLevel">University</meta>
    <meta name="contentLevel">Research</meta>
    <meta name="contentLevel">Amateur</meta>
```

Most of the attributes are easy to understand, see this page for detailed explanations and more meta elements.

Meta attribute "subject" is defined several times by different keywords defining data from UAT (Unified Astronomy Thesaurus). At least one of them must refer to a global topic listed in this page. In the context of VESPA, the 3 appropriate global topics are : "Exoplanet astronomy", "Solar physics" and "Solar system astronomy". The attribute "source" refers to the resource-related paper. Here, "contentLevel" takes the four values "General", "University", "Research", "Amateur" but it could take only some of these.

Table definition

Then, **<table>** definition starts, in every EPN-TAP services, the table **<id>** and **<mixin>** must respectively take the values "epn_core" and " //epntap2#table-2.0"**.**

spatial_frame_type attribute defines the type of coordinate system for the defined granules, it could take several values (listed in spatial_frame_type section of EPN TAP V2 parameter description) its choice will impact the coordinates definition (parameters c1, c2 and c3).

The mixin "//epntap2#table-2_0" provides a standard definition of mandatory parameters and some optional ones. Mandatory parameters will be automatically present in the table and you may specify predefined optional columns you want to include in **optional_columns** attribute. Then, additional columns could be added, but it is necessary to define them manually in **<column>** tags.

The tree optional columns *time_scale*, *publisher* and *bib_reference* are added to the table in this example.

```
   <table id="epn_core" onDisk="true" adql="True">

      <mixin spatial_frame_type="none"
      optional_columns= "time_scale publisher bib_reference" >//epntap2#table-2_0<
/mixin>
```

After mixin definition, you can start extra-parameters definition with the tag <column>. To do that, you should define the attributes **name**, **type**, **tablehead**, **unit** (if relevant, listed here), **description**, **ucd** (a set of keywords which defines the type of data, see ucd IVOA documentation) and **verblevel** (a rate under 30 defining the columns importance). After extra-columns are set, the table definition is complete.

```
      <column name="distance_to_primary" type="double precision"
      tablehead="Distance_to_primary" unit="km"
      description="Extra: Mean heliocentric distance (semi-major axis)"
      ucd="pos.distance;stat.min"
      verbLevel="2"/>
      <column name= ... />
      ...
      <column name= ... />

   </table>
```

Data ingestion

Data ingestion starts with the tag **<data>**. For the case of data imported from a CSV, **id** must take the value "import" and the CSV path must be indicated in **<sources>** element. **<csvGrammars>** for EPN-TAP contains global ingestion rules . The **<rowfilter>** defined here gives an automatic assignment to the CSV and table columns (mandatory, optional and added columns defined earlier) which have the same name. It is possible to define another rowfilter to add special ingestion rules (see an example here).

```
<data id="import">
   <sources>Masses2.csv</sources>
   <csvGrammar>
      <rowfilter procDef="//products#define">
         <bind name="table">"\schema.epn_core"</bind>
      </rowfilter>
   </csvGrammar>
```

Still into the <data> tag, the element <make table="epn_core"> aims to fill columns of the table epn_core.

The **<map>** elements associate values to the columns which has not been filled automatically by the rowfilter or need post-processing :

Constant value columns must be set like:

<map key="{column name}">{constant_value}</map>

Varying value columns could be defined by:

<map key="{EPN TAP column name}" source="{csv column name}"/>

or

<map key="{EPN TAP column name}" >{varying_value}</map>

Where {varying_value} could link to another column with the prefix @ or post-process it with simple operations in Python (e.g: *<map key="column">@column1+"text"+@column2[2:8]</map>*), this second method is not illustrated in the service planets. To make more complex post-processing, is it possible to use <code> tag into the rowfilter (see here for an example).

```
<make table="epn_core">
   <rowmaker idmaps="*">
      <map key="obs_id" source="obs_id" />
      ...
      <map key="measurement_type">"phys.mass#phys.size.radius"</map>
```

After map elements definition, the table construction is finished and the service could be published.

- ## Install and check tables

To install your service on your DaCHS server, you have to create a directory on your EPN-TAP server containing your RD and your CSV file into gavodachs directory : /var/gavo/inputs/{servicename}, where servicename is the same as <resource schema=".."> value into the RD.

For our example, connect to your server and, into the gavodachs directory, create the directory "planets" and a "data" subdirectory :

```
$ sudo mkdir /var/gavo/inputs/planets
```

Then go to the directory in which you have downloaded the RD and the CSV of the example "planets" from gitlab (here we assume its path is ~/planets) and copy these files into the directory previously created:

```
$ cd ~/planets
~/planets$ sudo cp q.rd /var/gavo/inputs/planets/q.rd
~/planets$ sudo cp Masses2.csv /var/gavo/inputs/planets/Masses2.csv
```

Then, go to your gavodachs resource directory

```
$ cd  /var/gavo/inputs/planets/
```

Now, you have to check the syntax of your RD with the command **dachs val** :

```
/var/gavo/inputs/planets/$ sudo dachs val q.rd
```

It returns *"q.rd – OK"* if the syntax of your RD is correct.

When the syntax is correct, you can import your service on your DaCHS server with the command **dachs imp** :

```
/var/gavo/inputs/planets/$ sudo dachs imp q.rd
```

If the service is correctly imported, the following message will appear

```
Making data planets/q#import
Starting /var/gavo/inputs/planets/data/Masses2.csv
Done /var/gavo/inputs/planets/data/Masses2.csv, read 8
Shipped 8/8
```

Then, you can restart the server to check the newly settled-up service with **dachs serve restart** :

```
/var/gavo/inputs/planets/$ dachs serve restart
```

The 3 steps (**dachs val** q.rd, **dachs imp** q.rd and **dachs serve restart**) are necessary each time the RD is modified.

After that, on your browser, type the address :

```
http://<<my_servername>>.<<my_domain>>/__system__/dc_tables/list
```

With <<my_servername>> and <<my_domain>> previously settled-up during your server installation. Click on *table info* of your service name, you can now check the list of columns and its description and metadata on the left. It is possible to send ADQL queries to interrogate the service.

For example, to select the whole planets database in an ADQL query, you can type:

```
SELECT * FROM planets.epn_core
```

See Checking your VESPA service for more information on checking your service.

- **Register the service**

To make your own service available on VESPA portal, you may first follow this tutorial : Registering your VESPA EPN-TAP Server. It consists in configuration and publication of your DaCHS server in the IVOA Registry of Registries (RoR), and publication of your TAP service. Finally, the VESPA team has to review this new service before making it available on VESPA Query Interface.

The service Planets may not be registered because it is already available in VESPA.

To go further ...

Another example and more detailed explanations on the setting up of an EPN-TAP service with iks service.

Another method to fill the table with a custom grammar using a python routine.

It is also possible to import data from MySQL or PostgreSQL databases in the resource descriptor using odbcgrammar .