Europlanet-RI / Europlanet Table Access Protocol Installation Tutorial

Draft 0.4
19/11/2012

EC Grant agreement n°228319

AUTHORS

Pierre Le Sidaner
DIO / OV, Observatoire de Paris
Stéphane Erard
LESIA, Observatoire de Paris / CNRS
Baptiste Cecconi
CDPP / LESIA, Observatoire de Paris / CNRS
Renaud Savalle
DIO / OV, Observatoire de Paris

Introduction

To attend the tutorial "publishing data in IDIS using EPN-TAP" you will need a laptop (or share one with your neighbor).
All the work will be performed on a server set up for the tutorial, so there is no specific software to install.
However, to access the server and to visualize the data locally, a few common tools will be useful:

* To access the server you will need to use ssh protocol
 - It's native on Linux & Mac OS X.
 - For Windows you can use putty available at:
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
 retrieve the programs putty.exe, pscp.exe and psftp.exe

* To transfer files if you are not familiar with the (s)ftp command line interface, you may install filezilla, available at:
 http://filezilla-project.org/download.php?type=client

* Then to retrieve and visualize the data we suggest that you use Topcat, Aladin and VOSpec.

To use those 3 clients you need a Java Virtual Machine, which is usually installed on every computer.

You need to download their installer or jar files. Test the jar file by clicking on the icon or by issuing the shell command:
java -jar “the_application.jar”
A window with the software will appear

* Aladin
http://aladin.u-strasbg.fr/java/nph-aladin.pl?frame=downloading

* Topcat

http://www.star.bris.ac.uk/~mbt/topcat/#install

* VOSpec
http://www.sciops.esa.int/index.php?project=ESAVO&page=vospec

There is no prerequisite on the knowledge of the participant about SQL and the POSTGRESQL database. In this case, the database is already built and a view called epn_core is almost complete.
DaCHS is a framework that provides the IVOA TAP service. That software is developed and maintained by Markus Demleitner from the German VO (GAVO). We will use that implementation, and install a service of atmospheric profiles on TITAN coming from the CASSINI spacecraft, preprocessed by Sandrine Vinatier from LESIA/Observatoire de Paris.
Since we are not sure to be able to perform the tutorial on line, we have installed a version of DaCHS on a local server and built a local network; this is called Solution 1. The server IP address will be given during the tutorial. Solution 2 is to use a remote server on line: voparis-cdpp.obspm.fr. On that server, the username is dal_user with password idis. The user gavo also has password idis.

Since it is a small database we will ask you to ingest the content of the database into the server. Every participant will create his own service inside the gavo database, and they need to be kept separated. The solution inside PostgreSQL is to create a schema for each user. We suggest you edit the file ingest_titan.sql and rename the schema with a specific name of your choice (only ascii character with no space and no . [dot])

1- Building your service database

a) Get the database ingestion script titandb.sql from the server
Solution 1
```
wget http://voparis-europlanet.obspm.fr/tutorial/titandb.sql
```
or get it with your web browser at this address
Solution 2
```
wget <<adress of the server given during tutorial>>:8080/titandb.sql
```
or get it with your web browser at this address

Once you have it

b) Save it as your own ingestion file, change the schema name and transfer it.
* you need to specify your schema name. It's the name of the place you will have your database stored into the server. It must be composed of ASCII characters without punctuation.

Edit titandb.sql and replace myschema by the name you have chosen on 5 occurrences:

1. *For the creation of the schema*

2. *For the creation of the two tables*
3. *For the data ingestion of the two tables*

The transfer to the server can be made by command line using sftp or filezilla

To ingest the database on the server:

Create the directory /var/gavo/inputs/<myschema>

You have to push titandb.sql using filezilla and transfer it as dal_user into the directory /var/gavo/inputs/<myschema>

then open a ssh session on the server (using solution 1 or 2) as dal_user

then ingest the database as gavo user, and replace <myschema> by the name you have used for your schema.

```
psql -U gavo gavo < /var/gavo/inputs/<myschema>/titandb.sql
```

The password is b55f4ce621964cc9f240

The output should be

```
BEGIN
CREATE SCHEMA
CREATE TABLE
INSERT 0 84
CREATE TABLE
INSERT 0 9
COMMIT
```

Those messages indicate that 2 tables were created
Ctrl-D to exit the shell

2- Creating the epn_core view

We want every service to have all mandatory parameters in the database with specific units and formats. However, we don’t want to ask providers to modify their databases. We therefore use a database functionality called a view (virtual table) that maps the fields of table(s) with a possible on the fly transform of units. Example: transform a date into Julian Day.

We don’t ask you to create the epn_core view from scratch. Some examples are available from http://voparis-europlanet.obspm.fr/implementation/examples
and all EPN-TAP mandatory fields are present in the view epn_core.

During the tutorial we provide you with the sql script to create the view

a) get the creation file for the epn_core view from the server

Solution 1

```
wget http://voparis-europlanet.obspm.fr/tutorial/view_case1.sql
```

or get it with your web browser at this address

Solution 2

```
wget <<adress of the server given during tutorial>>:8080/view_case2.sql
```

or get it with your web browser at this address

Once you have it
With Solution 2 you have to edit the file and give the address of the server for the field access_url.
In both cases you have to change the name of the schema to <myschema>.

To know more about what you are doing:
The view allows you to match some columns of the database with the epn-core view. This is the case for:
* longitude AS c1min. longitude is the column of the table and c1min is the specific term on epn_core
  But you can also change unit
* cast(to_char(date::date,'J') as double precision) AS t_min
this transforms the field “date” (ISO formatted) of the table into “t_min” (in Julian days) in the view.
* text 'granule' AS resource_type
this associates the string “granule” to the field “resource_type”. This allows not repeating this value for each row. It's written only once in the view, as a fixed value.
* NULL AS time_exp_min
this assigns the NULL value to non-relevant fields.

We make one view for dataset+granule for the two tables using UNION ALL to concatenate views using one SELECT command.

The transfer to the server can be made on the command line using sftp or filezilla

b) To create the view on the server:
You have to push view_case1.sql or view_case2.sql
Use filezilla and transfer it as dal_user in directory /var/gavo/inputs/<myschema>
Then open an ssh session on the server (using solution 1 or 2) as dal_user

Then ingest the database as gavo user , and replace myschema by the name of your schema
```
psql -U gavo gavo < /var/gavo/inputs/<myschema>/view_caseX.sql
```
The password is b55f4ce621964cc9f240
The result should be:
```
CREATE VIEW
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
```

These messages indicate that the view has been created.

You can try the view by entering the PSQL shell
```
psql -U gavo gavo
```
The password is b55f4ce621964cc9f240
and issuing the command
```
select * from gavo.<myschema>.epn_core;
```

The content of the view will appear.
Type ctrl-D to leave psql

3- Making the service available

The database is now completed. We have to provide DaCHS with the metadata of the view we want to make available to epn_tap. This will be made using a description file (called q.rd) and a command to ingest it.

The q.rd description file contains all the fields of the view with an associated description, plus metadata. In our case the metadata consist in the type of the data (integer, date, float …), and the units as taken from "Units in the VO" document available at http://nxg.me.uk/temp/VOUnits-1796.pdf or http://wiki.ivoa.net/twiki/bin/view/IVOA/VOUnitsRFC

Get the description file q.rd from the server

Solution 1
wget http://voparis-europlanet.obspm.fr/tutorial/q.rd
or get it with your web browser at this address
Solution 2
wget <<address of the server given during tutorial>>:8080/q.rd
or get it with your web browser at this address

Edit the q.rd file
Fill all the <ToDefine> by UCDs obtained from the following lists:

- http://dc.zah.uni-heidelberg.de/ucds/ui/ui/form
- http://cds.u-strasbg.fr/UCD/cgi-bin/descr2ucd
- http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.pdf

To understand what you're doing: the UCDs (Unified Contents Descriptors) represent the semantic information about the physical quantities contained in the columns of the views. They contain textual tokens separated by semi-columns, called words. Each word is composed of atoms separated by periods.

Modify the q.rd file. Change the name of the resource schema, replace <myschema> by the name of your schema.

Put the q.rd file into the directory you have created (/var/gavo/input/<myschema>) using sftp or filezilla
Then connect to ssh as dal_user

Check the validity of your q.rd file using the command:
```
gavo val q.rd
```
if the syntax is correct, it will answer
```
q.rd -- OK
```

then import the metadata
```
gavo imp -m q.rd
```

If import is successful, the output will indicate
```
Updating meta for import
```

When import is done, restart the DaCHS server:

```
sudo gavo serve stop
sudo gavo serve start
```

You can now test the service. Make a test query through the web browser using the following URL (replace serveraddress and myschema by the correct values):

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

4- Using the Europlanet client to query services

A client is a web form helping the user to write queries.
The client first retrieves a list of available services from a registry (yellow pages of declared services). In this case, the EPN-TAP client retrieves the service information endpoint from the VOPARIS registry, which harvests all the IVOA registries. The result of all services come asynchronously using Ajax technology.

Then the result can be sent directly to the client using SAMP:
the URL of the client is http://voparis-europlanet-new.obspm.fr/planetary/data/epn/query/all/
for your own service (not yet registered) use the menu “custom resource”
then fill the Resource URL and Schema name to test your service.

To test the database you have just defined:
time between 2004-01-01 and 2006-06-30, target name is titan and resource type is granule.

The client will transform your entries into an EPN-TAP query:

```
SELECT * FROM ... WHERE 1 = 1 AND target_name = 'titan' AND t_max = 2454281.5 AND t_min =
2453005.5 AND resource_type = 'granule'
```

You can also test access with the Topcat tool:
    open topcat
    chose menu VO->Table Access Protocol
    put in field TAP URL the url of your service
If you want the same result as the previous query

```
http://voparis-tap.obspm.fr/__system__/tap/run/tap
```

then click enter query

Topcat will use the specifications of TAP to browse the server and will propose you all the tables available for query
select Table : titan.epn_core from the combo box
All the fields from this table will appear.

You can see all of them in Topcat by typing the query in ADQL text field:

```
select * from titan.epn_core
```

type OK a table will be loaded
click on the 4th icon from the left in Topcat (the one looking like a grid)
The result of the query will appear 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
the result file is loaded into Topcat

select the icon “Scatter Plot”, a graph will appear. You can easily select the axes. The rainbow icon allows you to add a third column using colors.

5- Extra: registering a service

In order to use the IDIS client or Topcat to access the service, you must first register your service.
The IVOA or IDIS registies use an XML VOResource format. To simplify the work we provide you with a model file that you can modify: vopdc_obspm-lesia-epn-titan.xml available at
http://voparis-europlanet.obspm.fr/tutorial/vopdc_obspm-lesia-epn-titan.xml

The important thing to modify is the identifier. The identifier construction will be defined in a separate document (to be presented during the tutorial).

Services in IVOA registries depend on an Authority_ID. This “URI” is a construction describing your structure. It looks like ivo://my_observatory/my_laboratory/myteam
Then all the proposed services will derive from this naming authority ID. The service ID will be of the form:
ivo://my_observatory/my_laboratory/myteam/epn-myservice

Description of your services can also be made using the DaCHS registry functionalities. This is explained in the DaCHS documentation, especially:
`http://vo.ari.uni-heidelberg.de/docs/DaCHS/tutorial.html#starting-the-rd`