Fig. 1

Fig. 2

1. EPN-TAP via DaCHS

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

DaCHS: A VO publishing package
+
EPN-TAP: A standard schema for planetary data
+
Your Data
=
Almost instant interoperable data services

Links and stuff available from http://docs.g-vo.org/epntap.pdf

2. 4 Easy Steps to Data Publishing

(and one that's a little harder)

1. Install DaCHS[1]
2. Check out the DaCHS-EPN tutorial[2]
3. Starting from the example, adapt to your data
4. Play!
5. Later: Configure the registry interface, register your data

To show what's involved in "adapting" to your data, we'll have a look at what's in the example provided.

[1] `http://soft.g-vo.org/dachs`
[2] `http://docs.g-vo.org/DaCHS/tutorial.html#epn-tap`

3. The Resource Descriptor

Almost everything to do with publishing one resource with DaCHS is contained in a file called Resource Descriptor (RD):

- resource metadata
- table definitions
- ingestion rules
- service definitions [don't need that here]
- regression tests

The order actually doesn't matter to DaCHS, but it does to humans, so it's a good idea to stick to it. We will in this discussion.

Conventionally, resource descriptors reside in a subdirectory of DaCHS' input directory, one per resource ("data collection"). As long as there's just one of them, most people just call it `q.rd`. Let's go through the example.

4. Resource Metadata

This is mostly stuff useful to locate your data in registries or figure out more about it:

```
<meta name="creationDate">2014-08-15T13:00:00</meta>
<meta name="source">ftp://psa.esac.esa.int/....</meta>
<meta name="description">Images from the Rosetta flyby at asteroid
  (21) Lutetia.  Very few of them.  That's because this is just
  a fairly dumb example for how to use DaCHS to publish to EPN-TAP.
</meta>
<meta name="source">2010epsc.conf..664S</name>
<meta name="creator">
  Sierks, H.; Barbieri, C.; Koschny, D.; Lamy, P.; ...
</meta>
<meta name="subject">Asteroids</meta>
<meta name="subject">Spacecraft: Rosetta</meta>
<meta name="subject">Imaging</meta>
```

5. Table Definition

This is where you say how your table looks like. With EPN-TAP, you only have to define extra columns (you could overwrite existing columns if you don't like the description or want to add tricky things):

```
<table id="epn_core">
  <mixin
    processing_level="2"
    >//epntap#table</mixin>

  <column name="detector_temperature"
    unit="K" ucd="phys.temperature;instr"
    description="Temperature of the CCD"/>
</table>
```

The really magic part here is the mixin element. It says: "Put in all (standard) EPN-TAP columns at the start of this table." It also takes some global information on the table contents (processing level, units of the coordinates, etc).

6. Parsing stuff

DaCHS reads all kinds of data – these are in PDS format, so we use a PDS grammar and also explain that we have "products" (files) and that they have previews:

```
<data id="import">
  <property key="previewDir">previews</property>
  <sources pattern="data/*.IMG"/>
  <pdsGrammar>
    <rowfilter procDef="//products#define">
      <bind name="table">"\schema.epn_core"</bind>
      <bind name="mime">"image/x-pds"</bind>
      <bind name="preview_mime">"image/jpeg"</bind>
      <bind name="preview">\standardPreviewPath</bind>
    </rowfilter>
  </pdsGrammar>
```

Actually, we cheated a bit: pdsGrammar needs the exteral pyPDS[3] library to actually work, so you'll have to install this before the example works. As you can see, DaCHS accepts shell patterns when you tell it what files to read. As always with the elements used here, more details are available in the corresponding section in the reference documentation[4].

7. Mapping stuff

This is where the meat is:

```
<make table="epn_core">
  <rowmaker idmaps="*">
    <var key="exptime">float(@SR_ACQUIRE_OPTIONS[
        "EXPOSURE_DURATION"].split()[0])</var>

    <apply procDef="//epntap#populate" name="fillepn">
      <bind name="resource_type">"granule"</bind>
      <bind name="dataproduct_type">"im"</bind>
      <bind name="instrument_host_name">@INSTRUMENT_HOST_NAME</bind>
      <bind name="instrument_name">@INSTRUMENT_NAME</bind>
      <bind name="collection_id">@MISSION_PHASE_NAME</bind>
      <bind name="target_name">"(21) Lutetia"</bind>
...
```

The element content here are essentially Python expressions with the extra syntax that `@something` gives you the value of the something field in the input (in this case, the value in the header). `var` lets you assign variables you can use later, and the big apply here lets you say what goes where in the epn-tap columns (this lets us keep your mappings valid when EPN-TAP itself changes).

The hackery in exptime is necessary because pyPDS doesn't actually parse values in the metadata description. Anyone in for extending it?

[3] `https://github.com/RyanBalfanz/PyPDS/archive/master.zip`
[4] `http://docs.g-vo.org/DaCHS/ref.html#element-sources`

8. Do the import

`gavo imp q`

Except we've promised previews and DaCHS doesn't now how to make them, as it doesn't know the PDS format. Do it yourself in a processor:

```
class PreviewMaker(processing.PreviewMaker):
        def getPreviewData(self, accref):
                [...]
                imageData = f.read(imageDataLength)
                pixels = numpy.fromstring(
                        imageData, dtype=numpy.int16).reshape(
                          imageWidth, imageHeight)
                pixels[pixels<0] = 0
                return imgtools.jpegFromNumpyArray(
                  imgtools.scaleNumpyArray(pixels, 400))
```

`python bin/makePreview.py`

9. Done!

Start the server if necessary and query your table from TOPCAT or your web browser.

For test purposes, start the server with `gavo serve debug`, and the TAP access URL would be http://localhost:8080 as long as you don't configure something else.

Eventually: Register your data. DaCHS has all that's needed for that, but it's still a bit tedious to run a registry. Shortcuts are possible – talk to me.

10. Conclusion

Go forth and publish exciting data!