Abstract ideas from simple DAL protocols

From the developement of S3: a Simple protocol for theoretical data

Carlos Rodrigo Blanco[1]
Enrique Solano[1]
Miguel Cerviño[2]

[1] CAB,INTA-CSIC; SVO
[2] IAA, CSIC; SVO

IVOA interoperability meeting
Nara, Dec 6-11, 20010

History: Requirements

We wanted a protocol for theoretical data so that it was:

- Simple to develop.
  - The simpler the development of the service is, the more people will be willing to implement it $\Rightarrow$ more theoretical models in the VO.
- Similar to other simple protocols.
  - SIAP, SSAP...
- Flexible.
  - The relevant characteristics (parameters) can be very different for different models.

History: Generalization

Idea:

- take the SSAP protocol,
- get the abstract ideas in it,
- forget about restrictions referred to the particular type of data (spectra, ra, dec...).

How:

- $\sim$ 2005: TSAP (SSAP), for theoretical spectra
- $\sim$ 2008: S3, generalization for other types of theoretical data

Operations

Three main operations:

- Service description *(getCapabilities)*:
  - what queries can be done to the service? (valid parameters)
- Seach data query *(queryData)*:
  - Which results (files) are available for a given range of those parameters?
- Give me a particular file *(getData)*.

Operations: getCapabilities

What queries can be done to the service?

- What type of data is the service offering,
  - SSAP: spectra (time series?)
  - SIAP: images
  - theory: depends.
- Which parameters can be used for searching, and what values are allowed for each of them?
  - SSAP, SIAP...: predefined (POS, SIZE, BAND...)
  - theory: model dependent. Those specified by the service.

Operations: getCapabilities

Generalization:

- Don’t specify which parameters must/can be used for searching at DALI level.
- Describe how to write a generic getCapabilities answer.
    - parameters accepted in queries.
    - valid query values for those parameters.
    - query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)
- Specific protocols can add further restrictions for their particular case.
    - SSAP: POS, SIZE are required, etc.
    - ...

Operations: getCapabilities

Generalization:

Accepted parameters

<PARAM NAME="INPUT:myparam1"/>

<PARAM NAME="INPUT:myparam2"/>

- Describe how to write a generic getCapabilities answer.
  - parameters accepted in queries.
  - valid query values for those parameters.
  - query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)
- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

- Don't specify which parameters must/can be used for searching at DALI level.
- Describe how to write a generic getCapabilities answer.
  - parameters accepted in queries.
  - valid query values for those parameters.
  - query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)
- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

Valid values for the parameters

```
<PARAM NAME=”INPUT:myparam1”>
  <VALUES>
    <MIN value=”10”/>
    <MAX value=”100”/>
  </VALUES>
</PARAM>
```

l,
accepts ranges, accepts a list of values, etc.)

- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

Valid values for the parameters

```
<PARAM NAME=”INPUT:myparam1”>
  <VALUES type=”actual”>
    <OPTION value=”10”/>
    <OPTION value=”20”/>
    <OPTION value=”50”/>
    <OPTION value=”100”/>
  </VALUES>
</PARAM>
```

l,

- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

- Don't specify which parameters must/can be used for searching at DALI level.
- Describe how to write a generic getCapabilities answer.
  - parameters accepted in queries.
  - valid query values for those parameters.
  - query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)
- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

Query properties for the parameters

<PARAM NAME="INPUT:myparam1:<u>required</u>"/ >

<PARAM NAME="INPUT:myparam2:<u>list,range</u>"/ >

<PARAM NAME="INPUT:myparam3:<u>fixed</u>" VALUE="3"/ >

- parameters accepted in queries.
- valid query values for those parameters.
- query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)

- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: getCapabilities

Generalization:

- Don’t specify which parameters must/can be used for searching at DALI level.
- Describe how to write a generic getCapabilities answer.
  - parameters accepted in queries.
  - valid query values for those parameters.
  - query properties for those parameters (required, optional, accepts ranges, accepts a list of values, etc.)
- Specific protocols can add further restrictions for their particular case.
  - SSAP: POS, SIZE are required, etc.
  - ...

Operations: queryData

What results are available for given (range of) values for the accepted parameters?

The Query

- How to build the query
  - http://.../?param1=value1&param2=value1/value2...
  - params corresponding to the getCapabilities response. (or specified by particular DALI protocol)
- How to specify values, ranges, lists of values
  - range: param=value1/value2
  - list: param=value1,value2,value3

Operations: queryData

What results are available for given (range of) values for the accepted parameters?

The Answer

- INFO element (OK, ERROR...)
- Some PARAMS explaining the results
- A Table with the list of results:
  - A row for each result.
  - The needed fields.
  - One field for the AccessURL to the particular file(s) (when applicable).
- Particular protocols specify restrictions (required PARAMS, FIELDS, etc).

Operations: getData

Give me a particular file

- Usually
  - Just a URL to download a file (obtained in the queryData operation).
- Generalization
  - More than one file available for each result.
  - Ask for some preprocessing before downloading (change resolution, cutout, etc)
  - How to do this? ideas from theoretical case (simDAP)

getData+

- A result can have several *properties* (imagine them as different available tables)
    - In the typical case just one property (for instance: an spectrum)
- A *property* has one or more relevant *fields*
    - Spectrum, for instance: wavelength, flux
    - Image, for instance: x,y
    - Galaxy rotation curve: radius, velocity
    - ...
- This info can be given in the queryData answer in several ways or predefined by specific protocols (TBD)

getData+: Preprocessing

- Take the AccessUrl received in queryData.
  - http://.../?REQUEST=getData&FileID=100203
- Default case:
  - Use it to download the final file
- Multiple results: get one
  - http://.../?REQUEST=getData&FileID=100203&
    PROPERTY=spectrum&
- Or preprocess it:
  - http://.../?REQUEST=getData&FileID=100203&
    PROPERTY=spectrum&
    PREPROCESS=CUTOUT&
    FIELD=wavelength,1000,5000
- (syntax TBD)

Going further

- Seeing getCapabilities/queryData:
  - not just as operations requested
  - but **types of service answers**.
- If not REQUEST is specified in the query URL, the service decides what kind of answer is the right one (as a function of other posible parameters specified in the URL):
  - a getCapabilities answer
    - asking for values of some more parameters,
    - <RESOURCE type="capabilities">
  - a queryData answer
    - giving a list of available results,
    - <RESOURCE type="results">

Going further

This provides the option of much more flexibility (when needed)

- Allows for services with not rectangular data structure and more refined queries

a	b	c
1	0.5	3
1	0.7	5
2	0.1	—
2	0.2	—

$$\Longrightarrow \quad \begin{array}{lcl} a & \in & [1, 2] \\ b & \in & [0.1, 0.7] \\ c & \in & [3, 5] \end{array}$$

Going further

This provides the option of much more flexibility (when needed)

- Allows for services with not rectangular data structure and more refined queries

a	b	c
1	0.5	3
1	0.7	5
2	0.1	—
2	0.2	—

$\Longrightarrow$

a=1 $\Longrightarrow$

b	c
0.5	3
0.7	5

$\Longrightarrow$ $b \in [0.5, 0.7]$
$c \in [3, 5]$

a=2 $\Longrightarrow$

b
0.5
0.7

$\Longrightarrow$ $b \in [0.1, 0.2]$

Useful approach

This approach has been useful

- Includes SSAP, SIAP... main operations
  - each protocol adds its own restrictions and specific data models.
- Generalized:
  - theoretical spectra ($\sim$SSAP)
  - synthetic photometry for different models
  - isochrones, evolutionary tracks
  - complex asteroseismology models

THANK YOU!