Templating service URLs in DataLink service descriptors

A VOTable-appendix-like proposal

L.Michel, F.Bonnarel, P.Fernique

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

ESCAPE
European Science Cluster of Astronomy & Particle physics ESFRI research Infrastructures

Observatoire astronomique de Strasbourg

Rationale

- Services are dynamical. In that case :
    - Requested URL is partially built by inputs given by the user (or from tables rows).
        - Fixed URL solutions are not flexible enough (Too many combinations if several parameters)
- Current DataLink « service descriptor »
    - works well for « Param=value » buildup

Rationale

- Current DataLink « service descriptor »
  - Doesn't allow variability in root URL segments or non « name= value » parameters.
    - Specially « non VO » services. But not only (VOspace)
      - Example use cases:

        vos://cadc.nrc.ca~vault/pdowler/ivoa/virtual2020a/**[*from user*].**mp4a

        http://test.ivoa.net/tap/**[*sync or async*]**?query= « SELECT .... »

        using FIELD values to build URLs :

        – « name= value » with prefix/suffix in the value http:·......&Name===**[*from table*]**

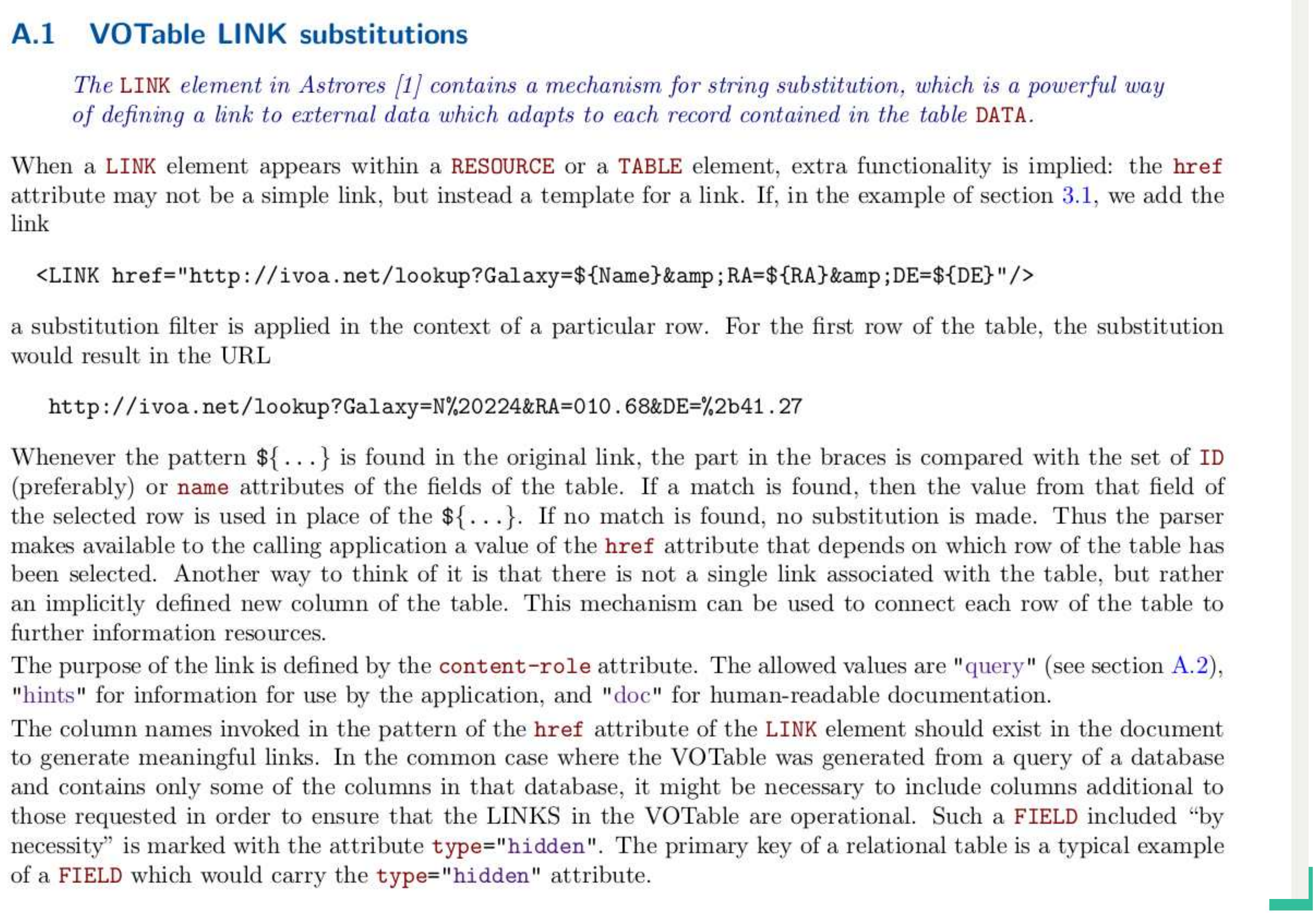
        – in root url segment http://www.eso.org/....../MGC**[*string from table]***B.fit.gz

        – Parameter without name « https:·....../Cat?**[*string from table*]** »

Solution(s)

- Goal : Explicitely referencing input params or fields into the accessURL URLs
- Based on RFC 6570?
  - Which part of the RFC to implement ?
  - Wildcarding, regular expressions ?
  - May become too complicated
  - Too too big investement for the client developpers
  - → we propose an alternative
- Based on VOTAble appendix A.1 (non normative)
  - Url templated by $*{field/param id}*
  - We propose to extend the mechanism from href in LINK to PARAMs such as « accessURL » PARAM in the service descriptor.
  - **!!!! Although non normative the LINK templating mechanism is extensively used by VizieR, simbad, LEDA, SVO, ESO, etc..**

VOTable LINK substitution



A.1 VOTable LINK substitutions

> *The* LINK *element in Astrores [1] contains a mechanism for string substitution, which is a powerful way of defining a link to external data which adapts to each record contained in the table* DATA.

When a LINK element appears within a RESOURCE or a TABLE element, extra functionality is implied: the href attribute may not be a simple link, but instead a template for a link. If, in the example of section 3.1, we add the link

```
<LINK href="http://ivoa.net/lookup?Galaxy=${Name}&amp;RA=${RA}&amp;DE=${DE}"/>
```

a substitution filter is applied in the context of a particular row. For the first row of the table, the substitution would result in the URL

```
http://ivoa.net/lookup?Galaxy=N%20224&RA=010.68&DE=%2b41.27
```

Whenever the pattern ${...} is found in the original link, the part in the braces is compared with the set of ID (preferably) or name attributes of the fields of the table. If a match is found, then the value from that field of the selected row is used in place of the ${...}. If no match is found, no substitution is made. Thus the parser makes available to the calling application a value of the href attribute that depends on which row of the table has been selected. Another way to think of it is that there is not a single link associated with the table, but rather an implicitly defined new column of the table. This mechanism can be used to connect each row of the table to further information resources.

The purpose of the link is defined by the content-role attribute. The allowed values are "query" (see section A.2), "hints" for information for use by the application, and "doc" for human-readable documentation.

The column names invoked in the pattern of the href attribute of the LINK element should exist in the document to generate meaningful links. In the common case where the VOTable was generated from a query of a database and contains only some of the columns in that database, it might be necessary to include columns additional to those requested in order to ensure that the LINKS in the VOTable are operational. Such a FIELD included “by necessity” is marked with the attribute type="hidden". The primary key of a relational table is a typical example of a FIELD which would carry the type="hidden" attribute.

How could it work (1) ?

- In AccessURL PARAM for the service
  - Use {inputParam:mode} to refer to a PARAM in the « inputPARAMS » GROUP

```
 <PARAM name="access_url" value="http://tapvizier.u-strasbg.fr/TAPVizieR/tap/${inputparam:mode}?>
<GROUP name="inputParams >
         <PARAM ID= »mode » name="mode" >
                <VALUES>
                    <OPTION value="sync">
                    <OPTION value="async">
                </VALUES>
         </PARAM>
         <PARAM name="REQUEST" >
                <VALUES>
                    <OPTION value="doQuery">
                </VALUES>
       </PARAM>
       ..............
 </GROUP>
```

How could it work (2) ?

- In AccessURL PARAM for the service
  - Root url built using value of FIELD with ID= « catid » in the main table

```
<PARAM name="access_url" value=" https://cdsarc.unistra.fr/viz-bin/cat/${catid} » />
<GROUP name="inputParams >
          <PARAM name= « format » />
        ..............
 </GROUP>
```

How could it work (3) ?

- In AccessURL PARAM for the service
  - Root url includes « parameter=value » template built using two FIELDS with ID=RAJ2000 and ID=DEJ2000

```
<PARAM name="access_url"
value="http://vizier.u-strasbg.fr/viz-bin/VizieR-5?-info=XML&-source=PFr&-
    c=${RAJ2000}${DEJ2000},rs=5 » />
```

How could it work (4) ?

- Special case with « parameter=value » using a single FIELD (eg ID=OBSID) can be refered in two ways :

```
<PARAM name="access_url" value="http://test.dummy.net/myservice&Ident=$
   {OBSID}» />
```

- Or like in DataLink 1.0 :

```
<PARAM name="access_url" value="http://test.dummy.net/myservice » />
<GROUP name= « inputParams » >
<PARAM name= « Ident » value= «  » ref= « OBSID » />
</GROUP>
```

Encoding

- In the charge of client for query elements
- not needed for path elements