1. TAPRegExt

Markus Demleitner, gavo@ari.uni-heidelberg.de

- Where does it go?
- Features that need description
- Elements currently proposed
- Just keys and values?

2. Where does it go?

TAPRegExt extends VOResource's capability type to describe TAP services. This is important in (at least) two places:

1. Registry records for TAP services
2. The content of the .../capabilities endpoint and the getCapabilities response of a TAP server

Indidentally, you can (and should!) already register TAP services *now*, even without TAPRegExt.

TAPRegExt should give ways to figure out properties of a service that are impossible to inspect from the outside using techniques specified in either UWS or TAP.

3. Concepts

At least the following concepts should be covered in a TAP service's capabilities:

- Data models (ObsCore!)
- Languages supported, user defined functions
- Output formats
- Upload methods
- Resource limits (match limit, upload limit)

Declaring the extent of geometry support (by language) would be nice, too.

4. Data Models

At least that's easy:

```
<dataModel ivo-id="ivo://ivoa.net/std/ObsCore-1.0"
  >ObsCore 1.0</dataModel>
```

i.e., ivo id for the machine, human-readable label (optional).

5. Languages

```
<language>
    <name>ADQL</name>
    <version>2.0</version>
    <description>ADQL 2.0</description>
    <userDefinedFunction>
      <signature
        >gavo_match(pattern TEXT, string TEXT) -&gt; INTEGER
        </signature>
      <description>gavo_match returns 1 if the POSIX regular
        expression pattern matches anything in string,
        0 otherwise.</description>
    </userDefinedFunction>
  </language>
```

Name is the value to pass to the LANG parameter, and multiple versions are allowed. Description is for human consumption.

User defined functions are a major pain. The solution proposed here assumes that machines will typically not need to access the data, though it's conceivable to parse the signature string.

The current proposal does not contain any provision for expression the level of support for geometry constructs.

6. Output Formats, Upload Methods

```
  <outputFormat>
    <mime>text/xml</mime>
    <description>VOTable, binary</description>
  </outputFormat>
  <uploadMethod
    ivo-id="ivo://ivoa.org/tap/uploadmethods#inline"/>
```

We use MIME hoping that clients can figure out if they can grok the format this way.

Upload methods are an (extensible) StandardsRegExt enumeration.

7. Resource Limits

```
  <retentionPeriod>
    <default>172800</default>
  </retentionPeriod>
  <executionDuration>
    <default>3600</default>
  </executionDuration>
  <outputLimit>
    <default unit="rows">2000</default>
    <hard unit="rows">20000000</hard>
  </outputLimit>
```

The unit on the data limits could also be bytes. A limit on upload sizes has a similar structure, but default probably makes no sense there.

8. or. . .

Quite a bit of TAPRegExt could be expressed using a single element containing a key-value-pair

```
<config key="geometrySupport.region">STC</config>
<config key="geometrySupport.contains">True</config>
```

– and it could be used to declare geometry support.

But it would be foreign to the rest of VOResource, not (practically) validateable by XSD, etc.

What do you think?