GWS prototypes:
UWS-1.1
VOSpace-2.1
VOSI-tables-1.1

Patrick Dowler

2015-06-15

National Research Council Canada
Conseil national de recherches Canada

UWS-1.1 Job Listing

- recently added support for job listing with filtering
  - used to always respond with a 403 (Forbidden)
  - anonymous listing still Forbidden
  - used in TAP and VOSpace
  - implementation is straightforward
  - we can only allow a user to see their own jobs – privacy!
- have not implemented AFTER
  - example timestamp value has a trailing Z; convention across the VO is that this should be UTC with no timezone indicator
- have not implemented LAST

UWS-1.1 Blocking

- added support for blocking via WAIT parameter
  - currently only block on phase change
  - there is a maximum blocking time before request will return even if job did not change
  - WAIT with no value waits for the maximum (~60 sec)
  - did not implement special WAIT=0, seems unnecessary for spec to bother with this
  - as an implementation detail, we have multiple load-balanced web servers (stateless) so “cannot” guarantee that *blocked* gets signals from *job executor*...
  - so: just simple escalating polling on server side

VOSpace-2.1

- added support to accept documents using 2.0 and 2.1 schema (nodes and transfers)
- nodes resource still emitting 2.0 documents
- changes are extra input details in transfer negotiation
  - specify securityMethod with each protocol
  - specify params that describe the transfer: content-length for pushToVoSpace

VOSpace-2.1 Transfer Negotiation

```
<vos:transfer xmlns:vos="http://www.ivoa.net/xml/VOSpace/v2.1">
<vos:target>vos://cadc.nrc.ca~vospace/pdowler/stuff.txt</vos:target>

  <vos:direction>pushToVoSpace</vos:direction>

  <vos:protocol uri="ivo://ivoa.net/vospace/core#httpsput">

    <vos:securityMethod uri="ivo://ivoa.net/sso#tls-with-certficate" />

  </vos:protocol>

  <vos:keepBytes>true</vos:keepBytes>

  <vos:param uri="ivo://ivoa.net/vospace/core#length">122079

   </vos:param>

</vos:transfer>
```

VOSpace-2.1 Transfer Negotiation

- clients should probably ask for all combinations of protocol and securityMethod they are willing/able to perform – including anonymous!
  - our distributed storage uses special transfer web service running at remote storage sites
  - these services cannot perform authorization check
  - we generate pre-authorized URLs that can be validated
  - result: for transfers we only actually support
  - HTTP
  - HTTPS with ivo://ivoa.net/sso#tls-with-certficate

VOSpace-2.1 Transfer Negotiation

- classic transfer negotiation involves an incoming and outgoing XML document
  - service can tell which version the client is speaking and respond with same (old 2.0 clients still work)
- shortcut param-based transfer
  - don't know what version the client expects
  - could respond with 2.0 if they don't specify securityMethod??
  - not yet fully implemented: we already operated with default behaviour of REQUEST=redirect – changing would break clients so software release coordination needed

VOSI-tables-1.1

- design goal was to define a RESTful resource tree since VOSI-tables is a simple hierarchy
  - /tables returns a <tableset>
  - /tables/$schema_name returns a <schema>
  - /tables/$schema_name/$table_name returns a <table>
- VOSI-tables xsd simply defines which elements from imported VODataService can be root element : it grows from 2 lines to 4!

VOSI-tables-1.1

```
<xsd:import namespace="http://www.ivoa.net/xml/VODataService/v1.1"
            schemaLocation="http://www.ivoa.net/xml/VODataService/v1.1" />

 <xsd:element name="tableset" type="vs:TableSet" />

  <!-- prototype root element for a single schema document -->
  <xsd:element name="schema" type="vs:TableSchema" />

  <!-- prototype root element for a single table document -->
  <xsd:element name="table" type="vs:Table" />
```

VOSI-tables-1.1

- compatibility goal: /tables should behave as in 1.0
- scalability goal: reduce the amount of output so it is manageable
- use optional parameters to limit depth of document
  - **detail=schema** to get <schema> but no <table>
  - **detail=table** to get <table> but no <column>
  - **/tables?detail=schema** (depth 1)
  - **/tables/tap_schema?detail=tables** (depth 1)
  - **/tables?detail=table** (depth 2 – everything but the columns)
    - single detail=min would probably suffice

VOSI-tables-1.1

- compatibility goal: /tables should behave as in 1.0
  - for services where this isn't feasible: 403 (Forbidden)?
- scalability goal: reduce the amount of output so it is manageable
  - REST + detail param solves this **if** content is organised such that individual <schema> are manageable
  - managable is quite large as <schema> and <table> don't contain much content (sans columns)
  - if organisation isn't enough, then: pagination
  - if you want to query, then: tap_schema