International
Virtual
Observatory
Alliance

UWS Registry Extension

Version 1.0

IVOA Working Draft 20180523

Working group
Grid and Web Services
This version
http://www.ivoa.net/documents/UWSRegExt/20180523
Latest version
http://www.ivoa.net/documents/UWSRegExt
Previous versions
This is the first public release
Author(s)
Brian Major, Markus Demleitner, Patrick Dowler, Mark Taylor
Editor(s)
Brian Major

Abstract

The Universal Worker Service (UWS) Registry Extension allows for extended metadata to be set on a VOResource `interface` for the purpose of discovering UWS job execution access URLs.

Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

Conformance-related definitions

The words "MUST", "SHALL", "SHOULD", "MAY", "RECOMMENDED", and "OPTIONAL" (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (Bradner, 1997).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

This note describes the registry extension for the Universal Worker Service (UWS) (Harrison and Rixon, 2010). Registry extensions add information to the schema defined in VOResource (Plante and Benson et al., 2008) that are specific to a particular standard, in this case UWS.

The UWS registry extension is very simple–it allows for two new values to be set on the *type* attribute in the *`interface`* element in a VOResource *`capability`*. These types have the purpose of identifying the *`interface`* as one of the synchronous or asynchronous UWS job execution features.

Although the UWS registry extension itself is simple, the history of the need for it is complicated. This history is described in section 2. The extension itself is described in section 3.

2 Background

2.1 Motivation

During the review of the TAP 1.1 specification ((Dowler and Rixon et al., 2010)), it was discovered that, when searching for TAP instances, there is the potential for clients of RegTAP ((Demleitner and Harrison et al., 2013) to discover multiple *interface* results per instance when only one is expected.

TAP clients currently follow the recommendations of RegTAP 1.1 to discover TAP access URLs and issue a query like this:

```
SELECT ivoid, access_url
    FROM rr.capability
    NATURAL JOIN rr.interface
    WHERE standard_id like 'ivo://ivoa.net/std/tap%'
    AND intf_type='vs:paramhttp'
```

This query is sufficient for most of the existing registered TAP services and will return only one result. Those records typically look like this:

```
<capability standardID="ivo://ivoa.net/std/TAP">
  <interface xsi:type="vod:ParamHTTP" role="std">
    <accessURL use="base">http://example.com/tap</accessURL>
  </interface>
</capability>
```

To then use the sync and async endpoints of the TAP service, clients simply append /sync and /async to the end of the base access URL, as per the recommendations of TAP version 1.0.

Two things have changed since VOResource version 1.03 that make this query to find all TAP services incorrect:

Change #1. Multiple standard IDs in TAP TAP 1.1 originally stated that the standard IDs of sync and async are distinct and each are registered. Although this gives services the flexibility to use URLs for sync and async that don't end in precisely /sync and /async, it breaks the pattern of TAP 1.0 service discovery. With the search recommended by RegTAP, clients would find the *interfaces* within each of the *capabilities* and attempt to append /sync and /async to the endpoints.

Here is an example of the *capability* element of the (now obsolete) synchronous endpoint in TAP 1.1:

```
<capability standardID="ivo://ivoa.net/std/TAP#sync-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
    <accessURL use="base">http://example.com/tap/sync</accessURL>
  </interface>
</capability>
```

Change #2. Multiple *interfaces* per *capability* Although, in VOResource, it has always been the case that multiple ***interfaces*** are allowed per ***capability***, no services had yet to be registered with such a configuration. This changed when some TAP implementations introduced TAP endpoints with a variety of ***SecurityMethods***. To document such a configuration, multiple ***interfaces*** per ***capability*** must be created, each with a different ***SecurityMethod***. Or, when the access URLs are the same, a single ***interface*** with multiple ***SecurityMethods*** could be created. Each configuration causes difficulties for registry service lookup.

Here is an example of the ***capability*** element of the synchronous endpoint in TAP 1.1 (now obsolete) with multiple ***interfaces*** to support multiple ***SecurityMethods***:

```
<capability standardID="ivo://ivoa.net/std/TAP#sync-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
    <accessURL use="base">http://example.com/tap/sync</accessURL>
  </interface>
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
    <accessURL use="base">https://example.com/tap/sync</accessURL>
    <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
  </interface>
</capability>
```

Note that, though this issue was discovered on the review of TAP 1.1, it applies to all IVOA services that need to provide potentially different access URLs for each ***SecurityMethod***.

2.2 Response

To address problem #2 described above, RegTAP 1.1 recommends that an additional constraint of having a null ***SecurityMethod*** is included in the where clause of the TAP service discovery query. However, for this to work, there cannot be a case where an ***interface*** is reused by multiple ***securityMethods***, as this would not allow the constraint of a null securityMethod to work if the endpoint supported other non-null securityMethods. So, VOResource 1.1 has ensures that at most one ***SecurityMethod*** is allowed per ***interface***.

Problem #1 is addressed in two ways: One: By removing multiple standard IDs from TAP 1.1; and, two: by the use of this UWS Registry Extension. TAP 1.1 has removed the use of standardIDs for sync and async and instead now only uses the version agnostic standardID from TAP 1.0: ***ivo://ivoa.net/std/TAP***. Clients can recognize the TAP service as version 1.1 by checking for a ***version*** attribute with the value "1.1" in the ***capability***. In order to allow services to choose their own access URLs

	Asynchronous	Synchronous
Interface Type	uws:Async	uws:Sync

Table 1: UWS Async and Sync interface types

for sync and async queries, *interface* elements within the *capability* are created for each type of *SecurityMethod* that is supported. There may be both sync and async versions of these *interfaces*. Whether an interface is sync or async is determined by the *type* attribute in the *interface*. To support TAP 1.0 backwards compatibility, a TAP 1.0 style *interface* must remain in the *capabilities*.

3 UWS Registry Extension

A number of IVOA services have functionality built on the Universal Worker Service. Jobs, which perform service-specific functions, can be created and started in either a synchronous or asynchronous manner. These services include TAP (Dowler and Rixon et al., 2010), VOSpace (Graham and Morris et al., 2009), and others. It is the UWS specification itself that defines how the lifecycle of these jobs is managed through a RESTful interface, not the services that are built on top of them. However, in the past, it has been left to the services to state how the various UWS endpoints (namely the synchronous and asynchronous job execution endpoints) are to be discovered through the IVOA registry.

This first version of the UWS Registry Extension enables UWS services to be registered with UWS-specific interface types so that they can be discovered by clients in a standard way defined by this note. This means that specifications built on UWS need not specify additional details on how to discover UWS RESTful endpoints.

The UWS Registry Extension introduces two new *interface* types: one for synchronous jobs and one for asynchronous jobs. The surrounding *capability* element should be marked with the minor version of service specification, as per the recommendations in the XML schema versioning note ((**?**))).

Future versions of the UWS Registry Extension may allow for more detail to be added about the UWS interfaces. This first version only supplies the two new *interface* types so that they can be used by UWS services to *tag* their sync and async endpoints.

3.1 Example TAP 1.1 capability

The following example describes how a TAP 1.1 service advertises its sync and async query endpoints under the single TAP capability. Both sync and

async support anonymous queries and authenticated queries with TLS (Grid and Web Services Working Group, 2008). The first two *interface* are there to provide backwards compatibility support for 1.0 TAP clients.

```
<capability standardID="ivo://ivoa.net/std/TAP">

  # TAP 1.0 support
  <capability standardID="ivo://ivoa.net/std/TAP">
    <interface xsi:type="vod:ParamHTTP" role="std">
      <accessURL use="base">http://example.com/tap</accessURL>
    </interface>
    <interface xsi:type="vod:ParamHTTP" role="std">
      <accessURL use="base">http://example.com/tap</accessURL>
      <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
    </interface>
  </capability>

  # TAP 1.1 support
  <interface
       xmlns:uws="http://www.ivoa.net/xml/UWSRegExt/v1.0"
       xsi:type="uws:Sync" role="std" version="1.1">
    <accessURL use="base">http://example.com/tap/sync</accessURL>
  </interface>
  <interface
       xmlns:uws="http://www.ivoa.net/xml/UWSRegExt/v1.0"
       xsi:type="uws:Sync" role="std" version="1.1">
    <accessURL use="base">https://example.com/tap/sync</accessURL>
    <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
  </interface>
  <interface
       xmlns:uws="http://www.ivoa.net/xml/UWSRegExt/v1.0"
       xsi:type="uws:Async" role="std" version="1.1">
    <accessURL use="base">http://example.com/tap/async</accessURL>
  </interface>
  <interface
       xmlns:uws="http://www.ivoa.net/xml/UWSRegExt/v1.0"
       xsi:type="uws:Async" role="std" version="1.1">
    <accessURL use="base">https://example.com/tap/async</accessURL>
    <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
  </interface>

</capability>
```

3.2 XML Schema

The following is the XML Schema Definition for the UWS Registry Extension version 1.0. It can be brought into an XML document's schema as a namespace with the URI *http://www.ivoa.net/xml/UWSRegExt/v1.0*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
    xmlns:vm="http://www.ivoa.net/xml/VOMetadata/v0.1"
    xmlns:uws="http://www.ivoa.net/xml/UWSRegExt/v0.1"
    version="1.0"
    targetNamespace="http://www.ivoa.net/xml/UWSRegExt/v0.1"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <xs:annotation>
        <xs:appinfo>
            <vm:schemaName>UWSregExt</vm:schemaName>
            <vm:schemaPrefix>xs</vm:schemaPrefix>
            <vm:targetPrefix>uws</vm:targetPrefix>
        </xs:appinfo>
        <xs:documentation>
            A description of the capabilities metadata for TAP services.
        </xs:documentation>
    </xs:annotation>

    <xs:import namespace="http://www.ivoa.net/xml/VOResource/v1.0"
        schemaLocation="http://www.ivoa.net/xml/VOResource/VOResource-v1.0.xsd"/>

    <xs:complexType name="Async">
        <xs:annotation>
            <xs:documentation>
            The UWS Async interface.
            </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="vr:Interface"/>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="Sync">
```

```
        <xs:annotation>
            <xs:documentation>
            The UWS Sync interface.
            </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="vr:Interface"/>
        </xs:complexContent>
    </xs:complexType>

</xs:schema>
```

A Changes from Previous Versions

No previous versions yet.

References

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.
http://www.ietf.org/rfc/rfc2119.txt

Demleitner, M., Harrison, P., Molinaro, M., Greene, G., Dower, T. and Perdikeas, M. (2013), 'IVOA registry relational schema', IVOA Working Draft.
http://www.ivoa.net/documents/RegTAP/

Dowler, P., Rixon, G. and Tody, D. (2010), 'Table access protocol version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/TAP

Graham, M., Morris, D. and Rixon, G. (2009), 'VOSpace specification', IVOA Recommendation.
http://www.ivoa.net/documents/VOSpace/

Grid and Web Services Working Group (2008), 'IVOA single-sign-on profile: Authentication mechanisms version 1.01'.
http://www.ivoa.net/documents/latest/SSOAuthMech.html

Harrison, P. and Rixon, G. (2010), 'Universal worker service pattern, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/UWS/20101010/

Plante, R., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G. and Stébé, A. (2008), 'VOResource: an XML encoding schema for resource metadata version 1.03', IVOA Recommendation. http://www.ivoa.net/documents/REC/ReR/VOResource-20080222.html