TAPRegExt: A VOResource Schema Extension for Describing TAP Services

Version 1.1

IVOA Working Draft 2015-11-17

Abstract

This document describes an XML encoding standard for metadata about services implementing the table access protocol TAP (Dowler et al. 2010), referred to as TAPRegExt. Instance documents are part of the service’s registry record or can be obtained from the service itself. They deliver information to both humans and software on the languages, output formats, and upload methods supported by the service, as well as data models implemented by the exposed tables, optional language features, and certain limits enforced by the service.
1 Introduction

The Table Access Protocol TAP (Dowler et al. 2010) allows VO clients to send queries to remote database servers and receive the results in standard formats. In addition, it defines means to discover database schemata on the remote side, to upload data from the local disk or third-party hosts, and more. TAP builds upon a variety of other standards, premier among which is the Universal Worker Service (Harrison & Rixon 2010), which describes how client and server can...
negotiate the execution of a query and the retrieval of results without having to maintain a continuous connection.

To accommodate a wide variety of requirements, the TAP specification offers implementors many choices on optional features, resource limits, or locally defined functionality. One purpose of TAPRegExt is to allow the service to communicate such choices to remote clients using the mechanisms laid down in the VO Service Interfaces standard (Grid and Web Services Working Group 2011)

Clients also need to discover TAP services offering certain kinds of data. Central to this is the concept of a registry in which resources can be described and consequently discovered by users and applications in the VO. Registries receive resource descriptions as defined in the IVOA standard (Plante et al. 2008). In this schema, support for a standard service protocol is described as a service’s capability; the associated metadata is contained within the service resource description’s <capability> element.

TAPRegExt defines this capability element for TAP services. In the context of registering TAP services, an important role filled by TAPRegExt is the communication of supported data models to the registry.

1.1 TAPRegExt within the VO Architecture

Figure 1: IVOA Architecture diagram with TAPRegExt and the related standards marked up
This specification directly relates to other IVOA standards in the following ways:

**VOResource, v1.03 (Plante et al. 2008)** Descriptions of services that support TAP are encoded using the VOResource XML schema. TAPRegExt is an extension of the VOResource core schema.

**TAP, v1.0 (Dowler et al. 2010)** The TAP standard defines some of the concepts that TAPRegExt deals with. The TAP standard document indirectly refers to this document in the specification of its capabilities endpoint. TAPRegExt 1.1 also reflects the evolution leading up to TAP 1.1.

**UWS, v1.0 (Harrison & Rixon 2010)** The UWS standard describes additional parameters the choices of which are communicated using TAPRegExt.

**StandardsRegExt (Harrison et al. 2012)** TAPRegExt uses the StandardKeyEnumeration mechanism introduced in StandardsRegExt to define controlled vocabularies.

This standard also relates to other IVOA standards:

**IVOA Support Interfaces, v1.0 (Grid and Web Services Working Group 2011)** VOSI describes the standard interfaces to discover metadata about services; this document defines the response TAP services should provide on the capabilities endpoint described by VOSI.

**IVOA defined data models** Data models specified by the IVOA can define the structure of database tables holding instances of those data models. The first examples of such definitions are ObsCore (Louys et al. 2011) and RegTAP (Demleitner et al. 2013). Services providing access to such tables declare that fact within TAPRegExt instance documents.

## 2 The Extension

### 2.1 The Schema Namespace and Location

The namespace associated with the TAPRegExt VOResource extension is

\[ \text{http://www.ivoa.net/xml/TAPRegExt/v1.0} \]

The namespace is unchanged from version 1.0 of this standard as no changes that could break clients are introduced.

Just like the namespace URI for the VOResource schema, the TAPRegExt namespace URI can be interpreted as a URL. Resolving it returns the XML schema document (also given as a part of this document in Appendix A) that defines the TAPRegExt schema.
Authors of VOResource instance documents may choose to provide a location for the VOResource XML schema document and its extensions using the `xsi:schemaLocation` attribute. While generators are free to provide any schema location (e.g., a local mirror), this specification recommends using the TAPRegExt namespace URI as its location URL, as in,

```
xsi:schemaLocation="http://www.ivoa.net/xml/TAPRegExt/v1.0
http://www.ivoa.net/xml/TAPRegExt/v1.0"
```

Note that you must give the `xsi:schemaLocation` of the TAPRegExt schema when the capability defined here is part of a published registry resource record as per the IVOA Registry Interface standard (Benson et al. 2009). This does not apply to the use in a TAP server’s capabilities endpoint.

2.2 Declaring Instantiated Data Models

The IVOA defines certain data models that can be instantiated in database tables exposed by a TAP service. This allows a query built exclusively on a data model or a set of data models to work on all TAP services exposing tables instantiating the data model(s).

In TAPRegExt, a data model is identified by its IVOA identifier (Plante et al. 2007). The example document in Appendix B uses this to define support for both the ObsCore and RegTAP data models.

`tr:DataModelType` Type Schema Documentation

An IVOA defined data model, identified by an IVOID intended for machine consumption and a short label intended for human consumption.

`tr:DataModelType` Type Schema Definition

```xml
<xs:complexType name="DataModelType" >
  <xs:simpleContent >
    <xs:extension base="xs:token" >
      <xs:attribute name="ivo-id" type="xs:anyURI" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

`tr:DataModelType` Attributes

ivo-id

- **Type**: a URI: `xs:anyURI`
- **Meaning**: The IVOID of the data model.
- **Occurrence**: required

2.3 Languages Supported

TAP services may offer a variety of query languages. In TAPRegExt, the `language` element allows the communication of what languages are available
on a service. TAP defines values of the LANG parameter to have either the form `<name>-<version>` or the form `<name>`, where the latter form leaves the choice of the version to the server. Therefore, a language is defined using a name and one or more versions.

The recommended way to associate larger amounts of documentation with a language entry in a capability element is via registration of the language using the mechanisms defined in StdRegExt (Harrison et al. 2012) and associating the registry record with the language element through the latter’s `ivo-id` attribute. The IVOID for the only language mandatory for TAP services, ADQL 2.0, is `ivo://ivoa.net/std/ADQL#v2.0`.

The type of the `ivo-id` attribute on version is `xs:anyURI` as opposed to `vr:IdentifierURI` since the latter does not allow fragment identifiers in VOReSource 1.0. The description constrains the value to be an IVOID (i.e., a URI with a schema of `ivo:`), though. The same reasoning applies to the `ivo-id` attributes of `outputFormat` and `uploadMethod`.

### `tr:Language` Type Schema Documentation

A query language supported by the service.

Each language element can describe one or more versions of a language. Either name alone or name-version can be used as values for the LANG parameter of the service.

### `tr:Language` Type Schema Definition

```xml
<xs:complexType name="Language">
  <xs:sequence>
    <xs:element name="name" type="xs:NCName" />
    <xs:element name="version" type="tr:Version" minOccurs="1" maxOccurs="unbounded" />
    <xs:element name="description" type="xs:token" minOccurs="0" />
    <xs:element name="languageFeatures" type="tr:LanguageFeatureList" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

### `tr:Language` Metadata Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Meaning</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>a prefixless XML name</td>
<td>The name of the language without a version suffix.</td>
<td><strong>required</strong></td>
</tr>
<tr>
<td><strong>version</strong></td>
<td>a string with optional attributes</td>
<td>A version of the language supported by the server.</td>
<td><strong>required; multiple occurrences allowed.</strong></td>
</tr>
<tr>
<td><strong>description</strong></td>
<td>string: <code>xs:token</code></td>
<td>A short, human-readable description of the query language.</td>
<td></td>
</tr>
</tbody>
</table>
Occurrence  optional
Element languageFeatures
  Type composite: tr:LanguageFeatureList
  Meaning Optional features of the query language, grouped by feature type.
  Occurrence  optional; multiple occurrences allowed.
  Comment  This includes listing user defined functions, geometry support, or similar concepts.

tr:Version Type Schema Documentation
One version of the language supported by the service.
If the service supports more than one version of the language, include multiple version elements. It is recommended that you use a version numbering scheme like MAJOR.MINOR in such a way that sorting by ascending character codes will leave the most recent version at the bottom of the list.

tr:Version Type Schema Definition
<x:simpleContent base="xs:token">
  <xs:attribute name="ivo-id" type="xs:anyURI" />
</xs:extension>
</xs:simpleContent>

tr:Version Attributes
ivo-id
  Type  a URI: xs:anyURI
  Meaning  An optional IVOID of the language.
  Occurrence  optional
  Comment  To more formally define a language supported by a service, a resource record for the language can be created, either centrally on the Registry of Registries or by other registry operators. When such a record exists, the ivo-id attribute of language should point to it.

Query languages may support optional features. For ADQL, the most prominent of those are user-defined functions, i.e., functions not defined in the language standard but added by the operators of the service, and geometry functions. Such optional features may be communicated to the service client in tr:languageFeatures elements.

Each such list is labelled with a type attribute indicating the type of language option being described. This string should be an IVOID whose semantics in this context, along with the semantics of the content of its descendant feature/form elements, can be documented in association with the language in question.

TAPRegExt itself defines the following feature types:
ivo://ivoa.net/std/TAPRegExt#features-udf

Each feature declares a user-defined ADQL (or similar) function supported. The content of the *form* element must be the signature of the function, written to match the signature nonterminal in the following grammar:

\[
\text{signature ::= \langle funcname \rangle \langle arglist \rangle \text{" ->"} \langle type\_name \rangle}
\]

\[
\text{funcname ::= \langle regular\_identifier \rangle}
\]

\[
\text{arglist ::= \langle ( \langle arg \rangle \{ \text{",} \langle arg \rangle \} \) \rangle \}
\]

\[
\text{arg ::= \langle regular\_identifier \rangle \langle type\_name \rangle}
\]

The *type\_name* nonterminal is not defined by the ADQL grammar in version 2.0. For the purposes of TAPRegExt, it is sufficient to assume it expands to "some sort of SQL type specifier" (which may include spaces and parentheses). For an enumeration of common types in ADQL, refer to the last column of the table in section 2.5 of the TAP standard (Dowler et al. 2010).

Example:

```xml
<languageFeatures type="ivo://ivoa.net/std/TAPRegExt#features-udf">
  <feature>
    <form>match(pattern TEXT, string TEXT) \rightarrow INTEGER</form>
    <description>
      match returns 1 if the POSIX regular expression pattern
      matches anything in string, 0 otherwise.
    </description>
  </feature>
</languageFeatures>
```

ivo://ivoa.net/std/TAPRegExt#features-adqlgeo

Each feature declares support for one of the geometry functions defined by ADQL (support for these functions is in general optional for ADQL implementations, though TAP imposes some constraints on what combinations of support are permitted).

The signature of these functions, where supported, is fixed by ADQL; the content of the *form* element is just the name of the function.

Example:

```xml
<feature>
  <form>CONTAINS</form>
</feature>
```

**tr:LanguageFeatureList** Type Schema Documentation

An enumeration of non-standard or non-mandatory features of a specific type implemented by the language.

A feature type is a language-dependent concept like “user defined function”, “geometry support”, or possibly “units supported”. A featureList gives all features of a given type applicable for the service. Multiple featureLists are possible.
All feature in a given list are of the same type. This type is declared using the mandatory type attribute, the value of which will typically be an IVOID. To see values defined in TAPRegExt, retrieve the ivo://ivoa.net/std/TAPRegExt resource record and look for keys starting with "features-".

**tr:LanguageFeatureList Type Schema Definition**

```xml
<xsd:complexType name="LanguageFeatureList" >
  <xsd:sequence >
    <xsd:element name="feature" type="tr:LanguageFeature" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="type" type="xsd:anyURI" use="required" />
</xsd:complexType>
```

**tr:LanguageFeatureList Attributes**

- **type**
  - **Type**: a URI: xs:anyURI
  - **Meaning**: The type of the features given here.
  - **Occurrence**: required
  - **Comment**: This is in general an IVOID. TAPRegExt itself gives IVOIDs for defining user defined functions and geometry support.

**tr:LanguageFeatureList Metadata Elements**

- **Element feature**
  - **Type**: composite: tr:LanguageFeature
  - **Meaning**: A language feature of the type given by the type attribute.
  - **Occurrence**: optional; multiple occurrences allowed.

**tr:LanguageFeature Type Schema Documentation**

A non-standard or non-mandatory feature implemented by the language.

**tr:LanguageFeature Type Schema Definition**

```xml
<xsd:complexType name="LanguageFeature" >
  <xsd:sequence >
    <xsd:element name="form" type="xs:token" />
    <xsd:element name="description" type="xs:string" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
```

**tr:LanguageFeature Metadata Elements**

- **Element form**
  - **Type**: string: xs:token
  - **Meaning**: Formal notation for the language feature.
  - **Occurrence**: required
  - **Comment**: The syntax for the content of this element is defined by the type attribute of its parent language list.
2.4 Output Formats

A TAP service may offer a variety of output formats. What output formats are available is defined using outputFormat elements. They declare an RFC 2046 media type (Freed & Borenstein 1996) as well as aliases (the shorthand forms the server also accepts in the FORMAT parameter). If desired, the format can be further described with an IVOID in the ivo-id attribute; TAPRegExt provides keys for some variants of VOTables which are not interoperably distinguishable by their MIME types so far:

- **output-votable-td** A VOTable in which all DATA elements contain a TABLE-DATA element
- **output-votable-binary** A VOTable in which all DATA elements contain a STREAM element with a BINARY child
- **output-votable-binary2** A VOTable in which all DATA elements contain a STREAM element with a BINARY2 child

### tr:OutputFormat Type Schema Documentation

An output format supported by the service.

All TAP services must support VOTable output, with media types as requested by the FORMAT parameter if applicable (cf. section 2.7.1 of the TAP standard).

The primary identifier for an output format is the RFC 2046 media type. If you want to register an output format, you must use a media type (or make one up using the x- syntax), although the concrete media syntax is not enforced by the schema.

For more detailed specification, an IVOID may be used.

### tr:OutputFormat Type Schema Definition

```xml
<xs:complexType name="OutputFormat" >
  <xs:sequence >
    <xs:element name="mime" type="xs:token" />
    <xs:element name="alias" type="xs:token" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="ivo-id" type="xs:anyURI" />
</xs:complexType>
```

### tr:OutputFormat Attributes

- **ivo-id**
  - **Type**: a URI: `xs:anyURI`
  - **Meaning**: An optional IVOID of the output format.
Comment When the media type does not uniquely define the format (or a
generic media type like application/octet-stream or text/plain is given),
the IVOID can point to a key or StandardsRegExt document defining
the format more precisely. To see values defined in TAPRegExt, retrieve
the ivo://ivoa.net/std/TAPRegExt resource record and look for keys
starting with "output-".

**tr:OutputFormat Metadata Elements**

**Element mime**

*Type* string: xs:token

*Meaning* The media type of this format.

*Occurrence* required

*Comment* The format of this string is specified by RFC 2046. The service
has to accept this string as a value of the FORMAT parameter.

**Element alias**

*Type* string: xs:token

*Meaning* Other values of FORMAT ("shorthands") that make the service
return documents with the media type.

*Occurrence* optional; multiple occurrences allowed.

### 2.5 Upload Methods

TAP services should allow the upload of VOTables. They can support various
methods to do this: HTTP upload, retrieval by URL, but also VOSSpace or
possibly retrieval using Grid protocols. Since an actual specification of the
details of such protocols is far beyond the scope of a registry document and
probably would not benefit clients anyway, the upload methods are given as
IVOIDs.

IVOIDs for the standard upload methods are provided within the resource
record ivo://ivoa.net/std/TAPRegExt. The IVOIDs are built by using the
keys as fragments after the TAPRegExt IVOID.

It is permitted to register upload methods under authorities other than
ivoa.net. The registry records can then provide more in-depth information.
For the upload methods defined in the TAP specification, however, the IVOIDs
of the keys in the TAPRegExt resource record must be used to enable clients
to identify supported methods using string comparisons.

This document defines the following protocol identifiers:

- **upload-inline** – HTTP upload as per section 2.5.2 of the TAP standard
  (Dowler et al. 2010).

- **upload-http** – retrieval from an http URL.

- **upload-https** – retrieval from an https URL.
• upload-ftp – retrieval from an ftp URL.

Thus, a service offering upload by retrieving from ftp and http URLs would say:

```xml
<uploadMethod ivo-id="ivo://ivoa.net/std/TAPRegExt#upload-http"/>
<uploadMethod ivo-id="ivo://ivoa.net/std/TAPRegExt#upload-ftp"/>
```

**tr:UploadMethod**  
*Type Schema Documentation*

An upload method as defined by IVOA. Upload methods are always identified by an IVOID. Descriptions can be obtained by dereferencing this IVOID. To see values defined in TAPRegExt, retrieve the ivo://ivoa.net/std/TAPRegExt resource record and look for keys starting with "upload-".

You can register custom upload methods, but you must use the standard IVOIDs for the upload methods defined in the TAP specification.

**tr:UploadMethod**  
*Type Schema Definition*

```xml
<xs:complexType name="UploadMethod">
  <xs:complexContent/>
  <xs:restriction base="xs:anyType">
    <xs:attribute name="ivo-id" type="xs:anyURI"/>
  </xs:restriction>
</xs:complexType>
```

**tr:UploadMethod**  
*Attributes*

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Meaning</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivo-id</td>
<td>a URI: xs:anyURI</td>
<td>The IVOID of the upload method.</td>
<td>optional</td>
</tr>
</tbody>
</table>

### 2.6 Resource Limits

TAP services usually impose certain limits on resource usage by clients, e.g., a maximum run time per query, or a maximum number of rows in the result set. Services assign such limits to newly created jobs and may allow raising the limits by means of queries or query parameters (e.g., the size of the result set is limited by the MAXREC parameter, whereas the date of job destruction may be changed by posting to the destruction parameter). Services may put some limit to how far the resource limitations may be raised.

TAPRegExt’s *capabilities* element allows the declaration of such limits. These declarations are primarily intended for human consumption and should give conservative guidelines. Thus, the operators of a service implementing a complex, possibly dynamic limits policy should give lower estimates here.

If a service supports authentication and has different limits depending on what user is authenticated, it should make an effort to guess the limits applying to a given client (e.g., when authentication tokens are present in the request).
Limits reported to the Registry should reflect limits for unauthenticated use unless the service does not admit unauthenticated requests.

The resource limits applying to newly created jobs are given in default elements, the limits beyond which users cannot raise the limits are given in hard elements.

Note that the absence of a specification of limits does not imply that no limits are enforced.

### 2.6.1 Limits on Time

This document defines two time-like resource limits:

- **retentionPeriod** – the time from job creation until destruction.
- **executionDuration** – the maximal run time given to a query.

All values in time-like limits are given in seconds. Both retentionPeriod and executionDuration are of type `tr:TimeLimits`.

**tr:TimeLimits Type Schema Documentation**

Time-valued limits, all values given in seconds.

**tr:TimeLimits Type Schema Definition**

```xml
<xs:complexType name="TimeLimits">
  <xs:sequence>
    <xs:element name="default" type="xs:integer" minOccurs="0" maxOccurs="1"/>
    <xs:element name="hard" type="xs:integer" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

**tr:TimeLimits Metadata Elements**

Element **default**

- **Type**: integer
- **Meaning**: The value of this limit for newly-created jobs, given in seconds.
- **Occurrence**: optional

Element **hard**

- **Type**: integer
- **Meaning**: The value this limit cannot be raised above, given in seconds.
- **Occurrence**: optional

### 2.6.2 Limits on Data

Limits on data are expressed much like time limits in that they give default and a hard value as well. Both those values have a unit attribute that can either be byte or row for data limits.

This document defines two resource limits on data:
• **outputLimit** – if *unit* is *row* here, the *default* gives the value of TAP’s MAXREC parameter the service will use when none is specified.

• **uploadLimit** – the maximum size of uploads. This is not a TAP adjustable parameter. The *default* value advises clients about the server’s wishes as to a limit above which some sort of acknowledgement should be requested from the user. The *hard* limit gives the maximum size of an upload to the server.

Data limits are defined using the *tr:DataLimits* and *tr:DataLimit* types:

*tr:DataLimits* Type Schema Documentation
Limits on data sizes, given in rows or bytes.

*tr:DataLimits* Type Schema Definition

```xml
<xs:complexType name="DataLimits" >
  <xs:sequence >
    <xs:element name="default" type="tr:DataLimit" minOccurs="0" maxOccurs="1" />
    <xs:element name="hard" type="tr:DataLimit" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>
```

*tr:DataLimits* Metadata Elements

Element *default*

Type an integer with optional attributes

*Meaning* The value of this limit for newly-created jobs.

*Occurrence* optional

Element *hard*

Type an integer with optional attributes

*Meaning* The value this limit cannot be raised above.

*Occurrence* optional

*tr:DataLimit* Type Schema Documentation
A limit on some data size, either in rows or in bytes.

*tr:DataLimit* Type Schema Definition

```xml
<xs:complexType name="DataLimit" >
  <xs:simpleContent >
    <xs:extension base="xs:integer" >
      <xs:attribute name="unit" use="required" >
        <xs:simpleType >
          <xs:restriction base="xs:token" >
            <xs:enumeration value="byte" />
            <xs:enumeration value="row" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```
2.7 The Capability Record

Using the types defined above, the `tr:TableAccess` type can be defined. Note that it is a type, not a (global) element. In instance documents, you will typically place it in a capability element with an explicit type specification, like this:

```xml
<capability
   xmlns:tr="http://www.ivoa.net/xml/TAP/v1.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   standardID="ivo://ivoa.net/std/TAP"
   xsi:type="tr:TableAccess">
   ...
</capability>
```

`tr:TableAccess`-typed capabilities can be used with arbitrary standard-IDs; this is different from previous TAPRegExt versions, which fixed it to `ivo://ivoa.net/std/TAP`. This URI should still be given for TAP 1.0 services. Services having other capabilities (e.g., TAP 1.1 or even UWS) will use other standardIDs.

**tr:TableAccess Type Schema Documentation**

The capabilities of a TAP server.
The capabilities attempt to define most issues that the TAP standard leaves to the implementors ("may", "should").

**tr:TableAccess Type Schema Definition**

```xml
<xs:complexType name="TableAccess" >
  <xs:complexContent >
    <xs:extension base="vr:Capability" >
      <xs:sequence >
        <xs:element name="dataModel" type="tr:DataModelType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="language" type="tr:Language" minOccurs="1" maxOccurs="unbounded" />
        <xs:element name="outputFormat" type="tr:OutputFormat" minOccurs="1" maxOccurs="unbounded" />
        <xs:element name="uploadMethod" type="tr:UploadMethod" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="retentionPeriod" type="tr:TimeLimits" minOccurs="0" maxOccurs="1" />
        <xs:element name="executionDuration" type="tr:TimeLimits" minOccurs="0" maxOccurs="1" />
        <xs:element name="outputLimit" type="tr:DataLimits" minOccurs="0" maxOccurs="1" />
        <xs:element name="uploadLimit" type="tr:DataLimits" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent >
</xs:complexType>
```
**tr:TableAccess Extension Metadata Elements**

**Element** `dataModel`  
- **Type**: a string with optional attributes  
- **Meaning**: Identifier of IVOA-approved data model supported by the service.  
- **Occurrence**: optional; multiple occurrences allowed.

**Element** `language`  
- **Type**: composite: `tr:Language`  
- **Meaning**: Language supported by the service.  
- **Occurrence**: required; multiple occurrences allowed.

**Element** `outputFormat`  
- **Type**: composite: `tr:OutputFormat`  
- **Meaning**: Output format supported by the service.  
- **Occurrence**: required; multiple occurrences allowed.

**Element** `uploadMethod`  
- **Type**: composite: `tr:UploadMethod`  
- **Meaning**: Upload method supported by the service.  
- **Occurrence**: optional; multiple occurrences allowed.  
- **Comment**: The absence of upload methods indicates that the service does not support uploads at all.

**Element** `retentionPeriod`  
- **Type**: composite: `tr:TimeLimits`  
- **Meaning**: Limits on the time between job creation and destruction time.  
- **Occurrence**: optional

**Element** `executionDuration`  
- **Type**: composite: `tr:TimeLimits`  
- **Meaning**: Limits on executionDuration.  
- **Occurrence**: optional

**Element** `uploadLimit`  
- **Type**: composite: `tr:DataLimits`  
- **Meaning**: Limits on the size of uploaded data.  
- **Occurrence**: optional

**Element** `outputLimit`  
- **Type**: composite: `tr:DataLimits`  
- **Meaning**: Limits on the size of data returned.  
- **Occurrence**: optional
A The Full Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xn="http://www.w3.org/2001/XMLSchema"
xmlns:xv="http://www.ivoa.net/xml/VOResource/v1.0"
xmlns:xm="http://www.ivoa.net/xml/VOMetadata/v0.1"
xmlns:tx="http://www.ivoa.net/xml/TAPRegExt/v1.0"
version="1.0"
targetNamespace="http://www.ivoa.net/xml/TAPRegExt/v1.0"
elementFormDefault="unqualified"
attributeFormDefault="unqualified"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<xs:annotation>
<xs:appinfo>
<vx:schemaName>TAPRegExt</vx:schemaName>
<vx:schemaPrefix>xn</vx:schemaPrefix>
<vx:targetPrefix>tx</vx: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="TableAccess">
<xs:annotation>
<xs:documentation>
The capabilities of a TAP server.
</xs:documentation>
<xs:documentation>
The capabilities attempt to define most issues that the
TAP standard leaves to the implementors ("may", "should").
</xs:documentation>
</xs:annotation>
<xs:complexContent>
<xs:extension base="vx:Capability">
<xs:sequence>
<xs:element name="dataModel" type="tx:DataModelType"
minOccurs="0" maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>
Identifier of IVOA-approved data model supported by the
service.
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="language" type="tx:Language"
minOccurs="1" maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>
Language supported by the service.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>
```
<xs:element name="outputFormat" type="tr:OutputFormat"
    minOccurs="1" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>
            Output format supported by the service.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="uploadMethod" type="tr:UploadMethod"
    minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>
            Upload method supported by the service.
        </xs:documentation>
        <xs:documentation>
            The absence of upload methods indicates
            that the service does not support uploads
            at all.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="retentionPeriod" type="tr:TimeLimits"
    minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>
            Limits on the time between job creation and
            destruction time.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="executionDuration" type="tr:TimeLimits"
    minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>
            Limits on executionDuration.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="outputLimit" type="tr:DataLimits"
    minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>
            Limits on the size of data returned.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="uploadLimit" type="tr:DataLimits"
    minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>
            Limits on the size of uploaded data.
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:complexType name="DataModelType">
  <xs:annotation>
    <xs:documentation>
      An IVOA defined data model, identified by an IVOID intended for machine consumption and a short label intended for human consumption.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xs:token">
      <xs:attribute name="ivo-id" type="xs:anyURI" use="required">
        <xs:annotation>
          <xs:documentation>
            The IVOID of the data model.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="Language">
  <xs:annotation>
    <xs:documentation>
      A query language supported by the service.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="xs:NCName">
      <xs:annotation>
        <xs:documentation>
          The name of the language without a version suffix.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="version" type="tr:Version" minOccurs="1" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>
          Each language element can describe one or more versions of a language. Either name alone or name−version can be used as values for the LANG parameter of the service.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
A version of the language supported by the server.
</xs:documentation>
</xs:annotation>
</xs:element>

<xsl:element name="description" type="xs:token"
minOccurs="0">
<xsl:annotation>
<xsl:documentation>
A short, human-readable description of the query language.
</xs:documentation>
</xsl:annotation>
</xs:element>

<xsl:element name="languageFeatures" type="tr:LanguageFeatureList"
minOccurs="0" maxOccurs="unbounded">
<xsl:annotation>
<xsl:documentation>
Optional features of the query language, grouped by feature type.
</xs:documentation>
</xsl:annotation>
<xsl:documentation>
This includes listing user defined functions, geometry support, or similar concepts.
</xs:documentation>
</xs:element>
</xs:sequence>
</xs:complexType>

<xsl:complexType name="Version">
<xsl:annotation>
<xsl:documentation>
One version of the language supported by the service.
</xs:documentation>
<xsl:documentation>
If the service supports more than one version of the language, include multiple version elements. It is recommended that you use a version numbering scheme like MAJOR.MINOR in such a way that sorting by ascending character codes will leave the most recent version at the bottom of the list.
</xs:documentation>
</xsl:annotation>

<xsl:simpleContent>
<xsl:extension base="xs:token">
<xsl:attribute name="ivo-id" type="xs:anyURI">
<xsl:annotation>
<xsl:documentation>
An optional IVOID of the language.
</xs:documentation>
</xsl:annotation>
</xs:attribute>
<xsl:documentation>
To more formally define a language supported by a service, a resource record for the language can be created, either centrally on the Registry of Registries or by other registry operators.
</xs:documentation>
</xsl:extension>
</xsl:simpleContent>
</xs:complexType>
When such a record exists, the ivo−id attribute of language should point to it.

A feature type is a language−dependent concept like "user defined function", "geometry support", or possibly "units supported". A featureList gives all features of a given type applicable for the service. Multiple featureLists are possible.

All feature in a given list are of the same type. This type is declared using the mandatory type attribute, the value of which will typically be an IVOID.

To see values defined in TAPRegExt, retrieve the ivo://ivoa.net/std/TAPRegExt resource record and look for keys starting with "features−".
A non-standard or non-mandatory feature implemented by the language...

Formal notation for the language feature. The syntax for the content of this element is defined by the type attribute of its parent language list.

The primary identifier for an output format is the RFC 2046 media type. If you want to register an output format, you must use a media type (or make one up using the x-syntype), although the concrete media syntax is not enforced by the schema.

All TAP services must support VOTable output, with media types as requested by the FORMAT parameter if applicable (cf. section 2.7.1 of the TAP standard).

The service has to accept this string as a value of the FORMAT parameter.

The format of this string is specified by RFC 2046.
<xs:element name="alias" type="xs:token"
minOccurs="0" maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>
Other values of FORMAT ("shorthands") that make the service return documents with the media type.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>

<xs:attribute name="ivo-id" type="xs:anyURI">
<xs:annotation>
<xs:documentation>
An optional IVOID of the output format.
</xs:documentation>
<xs:documentation>
When the media type does not uniquely define the format (or a generic media type like application/octet-stream or text/plain is given), the IVOID can point to a key or StandardsRegExt document defining the format more precisely. To see values defined in TAPRegExt, retrieve the ivo://ivoa.net/std/TAPRegExt resource record and look for keys starting with "output-".
</xs:documentation>
</xs:annotation>
</xs:attribute>
</xs:complexType>

<xs:complexType name="UploadMethod">
<xs:annotation>
<xs:documentation>
An upload method as defined by IVOA.
</xs:documentation>
</xs:annotation>
<xs:complexContent>
<xs:restriction base="xs:anyType">
<xs:attribute name="ivo-id" type="xs:anyURI">
<xs:annotation>
</xs:annotation>
</xs:attribute>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="UploadMethod">
<xs:annotation>
<xs:documentation>
An upload method as defined by IVOA.
</xs:documentation>
</xs:annotation>
<xs:complexContent>
<xs:restriction base="xs:anyType">
<xs:attribute name="ivo-id" type="xs:anyURI">
<xs:annotation>
</xs:annotation>
</xs:attribute>
</xs:restriction>
</xs:complexContent>
</xs:complexType>
<xs:element name="default" type="xs:integer"
  minOccurs="0" maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      The value of this limit for newly-created jobs, given in seconds.
    </xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
<xs:complexType name="TimeLimits">
  <xs:annotation>
    <xs:documentation>
      Time-valued limits, all values given in seconds.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="default" type="xs:integer"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          The value of this limit for newly-created jobs, given in seconds.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="hard" type="xs:integer"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          The value this limit cannot be raised above, given in seconds.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:complexType>
The value this limit cannot be raised above.

B Example Document

This appendix contains an instance document as it should be delivered from the VOSI capability endpoint. When embedded in a VOResource record, the capability elements would be direct children of the vr:Resource element.

```xml
<cap:capabilities
 xmlns:cap="http://www.ivoa.net/xml/VOSICapabilities/v1.0"
 xmlns:tr="http://www.ivoa.net/xml/TAPRegExt/v1.0"
 xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0"
 xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
 xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
 <capability
 standardID="ivo://ivoa.net/std/VOSI#availability"
 >
 <interface
 xsi:type="vs:ParamHTTP"
 >
 <accessURL use="full">http://localhost:8080/availability</accessURL>
 </interface>
 </capability>
</cap:capabilities>
```
standardID="ivo://ivoa.net/std/VOSI#capabilities">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://localhost:8080/capabilities</accessURL>
  </interface>
</capability>

<capability standardID="ivo://ivoa.net/std/VOSI#tables">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://localhost:8080/tableMetadata</accessURL>
  </interface>
</capability>

<capability standardID="ivo://ivoa.net/std/TAP" xsi:type="tr:TableAccess">
  <interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">http://localhost:8080/tap</accessURL>
  </interface>
  <dataModel ivo-id="ivo://ivoa.net/std/RegTAP#1.0">Registry 1.0</dataModel>
  <dataModel ivo-id="ivo://ivoa.net/std/ObsCore/v1.0">Obscore−1.0</dataModel>
  <dataModel ivo-id="ivo://org.gavo.dc/std/glots#tables-1.0">GloTS 1.0</dataModel>
</capability>

<language>
  <name>ADQL</name>
  <version ivo-id="ivo://ivoa.net/std/ADQL#v2.0">2.0</version>
  <description>ADQL 2.0</description>
  <languageFeatures type="ivo://ivoa.net/std/TAPRegExt#features-udf">
    <feature>
      <form>
        gavo_match(pattern TEXT, string TEXT) → INTEGER
      </form>
      <description>gavo_match returns 1 if the POSIX regular expression pattern matches anything in string, 0 otherwise.</description>
    </feature>
    <feature>
      <form>ivo_string_agg(expression TEXT, delimiter TEXT) → TEXT</form>
      <description>An aggregate function returning all values of expression within a GROUP concatenated with delimiter</description>
    </feature>
  </languageFeatures>
  <languageFeatures type="ivo://ivoa.net/std/TAPRegExt#features-adqlgeo">
    <feature>
      <form>BOX</form>
    </feature>
    <feature>
      <form>POINT</form>
    </feature>
    <feature>
      <form>CIRCLE</form>
    </feature>
    <feature>
      <form>POLYGON</form>
    </feature>
    <feature>
      <form>REGION</form>
    </feature>
    <feature>
      <form>CENTROID</form>
    </feature>
  </languageFeatures>
</language>
<feature>
  <form>COORD1</form>
</feature>
<feature>
  <form>COORD2</form>
</feature>
<feature>
  <form>DISTANCE</form>
</feature>
<feature>
  <form>CONTAINS</form>
</feature>
<feature>
  <form>INTERSECTS</form>
</feature>
<feature>
  <form>AREA</form>
</feature>
</languageFeatures>
</language>
<outputFormat ivo-id="ivo://ivoa.net/std/TAPRegExt#output-votable-binary">
  <mime>text/xml</mime>
</outputFormat>
<outputFormat>
  <mime>text/html</mime>
  <alias>html</alias>
</outputFormat>
<outputFormat ivo-id="ivo://ivoa.net/std/TAPRegExt#output-votable-binary2">
  <mime>application/x-votable+xml;serialization=binary2</mime>
  <alias>votable/b2</alias>
</outputFormat>
<outputFormat>
  <mime>application/fits</mime>
  <alias>fits</alias>
</outputFormat>
<outputFormat>
  <mime>text/csv</mime>
</outputFormat>
<outputFormat>
  <mime>text/csv;header=present</mime>
  <alias>csv</alias>
</outputFormat>
<outputFormat ivo-id="ivo://ivoa.net/std/TAPRegExt#output-votable-td">
  <mime>application/x-votable+xml;serialization=tabledata</mime>
  <alias>votable/td</alias>
</outputFormat>
<outputFormat>
  <mime>application/json</mime>
  <alias>json</alias>
</outputFormat>
<outputFormat ivo-id="ivo://ivoa.net/std/TAPRegExt#output-votable-binary">
  <mime>application/x-votable+xml</mime>
  <alias>votable</alias>
</outputFormat>
<outputFormat>
  <mime>text/plain</mime>
</outputFormat>
C    Changes from Previous Versions

C.1 Changes from WD-20110127

- userDefinedFunction was generalized to feature within languageFeatures.

- The uploadmethods StandardKeyEnumeration was replaced by a resource record for TAPRegExt as a whole. This now includes keys of output formats and features as well; therefore, upload method names in their new IVOIDs are prefixed with upload-

- Schema version was bumped to 1.0 (yes, we indulge in unversioned schema changes before this becomes REC).

- uploadLimit interpretation was changed: The default limit is now "advisory" and to be interpreted as such by clients, the hard limit is what is actually required by the server.

- There’s now an optional ivo-id attribute on the version element within language.

- There’s now an optional ivo-id attribute on output formats.

C.2 Changes from WD-20110727

- The namespace in the schema is now http://www.ivoa.net/xml/TAPRegExt/ v1.0 consistent with what has already been stated in the text.
• The IVOID for ADQL is now ivo://ivoa.net/std/ADQL#v2.0; it is defined here to be in ADQL’s record since we do not want to wait for the ADQL standard to be fixed, but ADQL versioning should really not be done here, so a TAPRegExt IVOID is out of the question.

• The IVOID of the TAPRegExt standard is now ivo://ivoa.net/std/TAPRegExt to conform with other standard IVOIDs. Unfortunately, this changes all other IVOIDs dependent on this.

• We now allow AnyURI on the ivo-id of language to allow fragment identifiers as, e.g., in ADQL.

C.3 Changes from PR-20120812

• Fixed units in limits to "row" and "byte".

C.4 Changes from REC-1.0

• No longer restricting TableAccess’ standardID to the (old) TAP standardID. TAPRegExt capabilities are now allowed with arbitrary standardIDs

• Changed former use of the term “IVORN” to “IVOID” (or Registry part, as appropriate) to comply with a proposed clarification of terminology in Identifiers 2.0

• The example now shows an entire capabilities response

• Repaired obscure data model URI in the example

• dataModel/@ivo-id is now typed xs:anyURI to allow fragment identifiers on it

• Migrated to ivoatex

References

URL: http://www.ivoa.net/documents/RegistryInterface/

URL: http://www.ivoa.net/documents/RegTAP/

URL: http://www.ivoa.net/documents/TAP
URL: http://www.ietf.org/rfc/rfc2046.txt

URL: http://www.ivoa.net/documents/VOSI/index.html

URL: http://www.ivoa.net/documents/StandardsRegExt/20120508/REC-Stan-dardsRegExt-1.0-20120508.html

URL: http://www.ivoa.net/documents/UWS

URL: http://www.ivoa.net/documents/ObsCore/20111028/REC-ObsCore-v1.0-20111028.pdf

URL: http://www.ivoa.net/documents/REC/ReR/VOResource-20080222.html

URL: http://www.ivoa.net/documents/REC/Identifiers/Identifiers-20070302.html