



TAPRegExt: a VOResource Schema Extension for Describing TAP Services Version 0.1

IVOA Working Draft 27 January 2011

This version:

<http://www.ivoa.net/Documents/TAPRegExt/WD-TAPRegExt-0.1-20100127.html>

Latest version:

<http://www.ivoa.net/Documents/TAPRegExt/>

Previous versions:

N/A

Working Group:

<http://www.ivoa.net/twiki/bin/view/IVOA/IvoaWGResReg>

Author(s):

[Markus Demleitner](#)

[Patrick Dowler](#)

[Ray Plante](#)

[Guy Rixon](#)

[Mark Taylor](#)

Abstract

This document describes an XML encoding standard for metadata about services implementing the table access protocol TAP [[TAP](#)], 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, user defined functions per language, and certain limits enforced by the service.

Status of this Document

The first release of this document was 2011 Jan 27.

This is an IVOA Internal 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".

Acknowledgments

This document has been developed with support from the German Astronomical Virtual Observatory (BMBF Bewilligungsnummer 05A08VHA). FILL-IN: Anyone else wants to be here?

This document borrows extensively from the StandardsRegExt [\[SRE\]](#) document.

Syntax Notation Using XML Schema

The eXtensible Markup Language, or XML, is document syntax for marking textual information with named tags and is defined by the World Wide Web Consortium (W3C) Recommendation XML 1.0 [\[XML\]](#). The set of XML tag names and the syntax rules for their use is referred to as the document schema. One way to formally define a schema for XML documents is using the W3C standard known as XML Schema [\[XSD\]](#).

This document defines the TAPRegExt schema using XML Schema. The full Schema document is listed in [Appendix A](#). Parts of the schema appear within the main sections of this document; however, documentation nodes have been left out for the sake of brevity.

Reference to specific elements and types defined in the VOResource schema include the namespaces prefix, `vr`, as in `vr:Resource` (a type defined in the VOResource schema). Reference to specific elements and types defined in the TAPRegExt schema include the namespaces prefix, `tr`, as in `tr:TableAccess` (a type defined in the TAPRegExt schema). Use of the `tr` prefix in compliant instance documents is strongly recommended, particularly in the applications that involve IVOA Registries. Elsewhere, the use is not required.

Contents

- 1 [Introduction](#)
 - 1.1 [Example Document](#)
 - 1.2 [Dependencies on other IVOA Standards](#)
- 2 [The `tr:TableAccess` Extension](#)
 - 2.1 [The Schema Namespace and Location](#)
 - 2.2 [Declaring Instantiated Data Models](#)
 - 2.3 [Languages Supported](#)
 - 2.4 [Output Formats](#)
 - 2.5 [Upload Methods](#)
 - 2.6 [Resource Limits](#)
 - 2.7 [The Capability Record](#)

1 Introduction

The Table Access Protocol TAP (see [\[TAP\]](#)) allows VO clients to send queries to remove 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 (see [\[UWS\]](#)), 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.

The TAP specification has intentionally been written to accommodate a wide variety of requirements. In consequence, implementors have to make a number of choices relevant to remote clients -- here understood to be software agents -- or their users. In order to communicate these choices in a standardized way, TAP services support querying for their capabilities, either by using HTTP GET on the capabilities child of the service's root resource (cf. [\[VOSI\]](#)), or by querying the sync endpoint with **REQUEST=getCapabilities**. This document defines the format of the document returned for either query.

The capability element defined here is also used when registering TAP services at an IVOA registry (see [\[REGWG\]](#)). As such it is analogous to the capability definitions contained in the extension schemas for SIA, SCS, SSA, and so on (available at [\[SCHEMA\]](#)). It is therefore defined as an XML Schema type derived from `vr:Capability` by first fixing the `standardID` attribute to `ivo://ivoa.net/std/TAP` and then extending it to give TAP-specific properties as described below.

1.1 Example Document

As an example, here is an instance document as it might be returned from a VOSI capability endpoint:

```
<?xml version="1.0"?>
<capability
  xmlns:tr="http://www.ivoa.net/xml/TAP/v0.1"
  xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
  xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  standardID="ivo://ivoa.net/std/TAP"
  xsi:type="tr:TableAccess">
  <interface role="std"
    xsi:type="vs:ParamHTTP">
    <accessURL use="base">http://localhost:8080/__system__/tap/run/tap</accessURL>
  </interface>
  <dataModel ivo-id="ivo://ivoa.net/std/ObsCore-1.0">ObsCore 1.0</dataModel>
  <language>
    <name>ADQL</name>
    <version>2.0</version>
    <description>ADQL 2.0</description>
    <userDefinedFunction>
      <signature>gavo_match(pattern TEXT, string TEXT) -> INTEGER</signature>
      <description>gavo_match returns 1 if the POSIX regular expression pattern matches anything in string, 0 otherwise.</description>
    </userDefinedFunction>
  </language>
</capability>
```

```

    </userDefinedFunction>
</language>
<outputFormat>
  <mime>text/xml</mime>
  <description>VOTable, binary</description>
</outputFormat>
<outputFormat>
  <mime>application/x-votable+xml;encoding=tabledata</mime>
  <alias>votable/td</alias>
  <description>VOTable, tabledata</description>
</outputFormat>
<outputFormat>
  <mime>text/html</mime>
  <alias>html</alias>
  <description>HTML table</description>
</outputFormat>
<outputFormat>
  <mime>application/fits</mime>
  <alias>fits</alias>
  <description>FITS binary table</description>
</outputFormat>
<outputFormat>
  <mime>text/csv</mime>
  <description>CSV without column labels</description>
</outputFormat>
<outputFormat>
  <mime>text/csv;header=present</mime>
  <alias>csv</alias>
  <description>CSV with column labels</description>
</outputFormat>
<outputFormat>
  <mime>text/tab-separated-values</mime>
  <alias>tsv</alias>
  <description>Tab separated values</description>
</outputFormat>
<outputFormat>
  <mime>application/x-votable+xml</mime>
  <alias>votable</alias>
  <description>VOTable, binary</description>
</outputFormat>
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#inline" />
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#http" />
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#https" />
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#ftp" />
<retentionPeriod>
  <default>172800</default>
</retentionPeriod>
<executionDuration>
  <default>3600</default>
</executionDuration>
<outputLimit>
  <default unit="rows">2000</default>
  <hard unit="rows">20000000</hard>
</outputLimit>
</capability>

```

1.2 Dependencies on other IVOA Standards

This specification relies directly on other IVOA standards in the following ways:

VOResource, v1.03 [\[VOR\]](#)

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 [\[TAP\]](#)

The TAP standard describes some of the concepts the declaration of which is the scope of TAPRegExt.

UWS, v1.0 [\[UWS\]](#)

The TAP standard describes additional concepts the declaration of which is the scope of TAPRegExt.

StandardsRegExt [\[SRE\]](#)

TAPRegExt uses the StandardKeyEnumeration mechanism introduced in StandardsRegExt to define controlled vocabularies.

This standard also relates to other IVOA standards:

IVOA Standard Interface, v1.0 [\[VOSI\]](#)

VOSI describes the standard interfaces to discover metadata about services; this document describes the response TAP services should provide on the `capabilities` endpoint described by VOSI.

IVOA defined data models

The IVOA defines data models database tables can conform to in order to facilitate cross-service queries. The first example of those is [\[ObsTAP\]](#). The support for queries against such data models is noted within TAPRegExt instance documents.

2 The `tr:TableAccess` Extension

2.1 The Schema Namespace and Location

The namespace associated with TAPRegExt VOResource extensions will be `http://www.ivoa.net/xml/TAPRegExt/v1.0`. Just like the namespace URI for the VOResource schema, the TAPRegExt namespace URI can be interpreted as a URL. Resolving it will return the XML Schema document (given 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 the author is free to choose any schema location, this specification recommends using the TAPRegExt namespace URI as its location URL (as illustrated in the example above), as in,

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

Note

The IVOA Registry Interface standard [\[RI\]](#) actually *requires* that the VOResource records it shares with other registries provide location URLs via `xsi:schemaLocation` for the VOResource schema and all legal extension schemas that are used in the records. This rule would apply to the TAPRegExt schema.

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 table(s) instantiating the data model(s).

In TAPRegExt, a data model is identified by its IVOA id. The first example for such a data model is ObsTAP (see [\[OT\]](#)).

tr:DataModelType Type Schema Definition

```
<xs:complexType name="DataModelType" >
  <xs:simpleContent >
    <xs:extension base="xs:token" >
      <xs:attribute name="ivo-id" type="vr:IdentifierURI" />
    </extension>
  </simpleContent>
</complexType>
```

tr:DataModelType Attributes	
Attribute	Definition
ivo-id	<i>Value type:</i> an IVOA Identifier URI: vr:IdentifierURI <i>Semantic Meaning:</i> The IVORN of the data model. <i>Occurrences:</i> optional

2.3 Languages Supported

TAP services may offer a variety of query languages. What languages are available is defined using the `language` element. 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. All compliant TAP services must at least declare ADQL.

tr:Language Type Schema Definition

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

tr:Language Metadata Elements	
Element	Definition

tr:Language Metadata Elements	
Element	Definition
name	<p><i>Value type:</i> <code>xs:NCName</code></p> <p><i>Semantic Meaning:</i> The name of the language without a version suffix.</p> <p><i>Occurrences:</i> required</p>
version	<p><i>Value type:</i> string: <code>xs:token</code></p> <p><i>Semantic Meaning:</i> One version of the language supported by the service.</p> <p><i>Occurrences:</i> required; multiple occurrences allowed.</p> <p><i>Comments:</i> 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.</p>
description	<p><i>Value type:</i> string: <code>xs:token</code></p> <p><i>Semantic Meaning:</i> A short, human-readable description of the query language.</p> <p><i>Occurrences:</i> optional</p>
userDefinedFunction	<p><i>Value type:</i> composite: tr:UserDefinedFunction</p> <p><i>Semantic Meaning:</i> Nonstandard functions available on this service.</p> <p><i>Occurrences:</i> optional; multiple occurrences allowed.</p>

Many query languages, first and foremost ADQL, have the notion of user defined functions, i.e., functions not defined in the ADQL standard but added by the operators of the service. A standard way to communicate those to the user is via the `tr:UserDefinedFunction` type.

Specifying function names, arguments, their types and return value(s) in a structured form is at least hard, in particular if the formalism should work for languages other than ADQL (which may have optional or named arguments, complex return types, etc). Therefore, function signatures are defined using a simple plain text representation. For ADQL, the content of the signature element must match the `signature` nonterminal in the following grammar:

```
signature ::= <funcname> <arglist> "-" <type_name>
funcname ::= <regular_identifier>
arglist ::= "(" <arg> { "," <arg> } ")"
arg ::= <regular_identifier> <type_name>
```

`regular_identifier` is defined in [\[ADQL\]](#), and `type_name` is one of the type specifiers from the table in Section 2.5 of [\[TAP\]](#). Other languages may need different formalisms, but authors are encouraged to maintain the character of this definition as closely as possible to help users familiar with the ADQL definitions make sense of the conventions

employed.

tr:UserDefinedFunction Type Schema Definition

```
<xs:complexType name="UserDefinedFunction" >
  <xs:sequence >
    <xs:element name="signature" type="xs:token" />
    <xs:element name="description" type="xs:string" minOccurs="0" />
  </sequence>
</complexType>
```

tr:UserDefinedFunction Metadata Elements	
Element	Definition
signature	<i>Value type:</i> string: xs:token <i>Semantic Meaning:</i> The function's signature. <i>Occurrences:</i> required <i>Comments:</i> The signature of a function in string form as defined in the TAPRegExt recommendation.
description	<i>Value type:</i> string: xs:string <i>Semantic Meaning:</i> Human-readable freeform documentation for the user defined function. <i>Occurrences:</i> optional

2.4 Output Formats

TAP service may offer a variety of output formats. What output formats are available is defined using the `outputFormat` element. They declare a MIME type (see [\[RFC2045\]](#)) as well as aliases (shorthand forms the server also accepts in the `FORMAT` parameter) and a short, human readable description that gives implementors a place to put further information on a given output format.

tr:OutputFormat Type Schema Definition

```
<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:element name="description" type="xs:token" minOccurs="0" maxOccurs="1" />
  </sequence>
</complexType>
```

tr:OutputFormat Metadata Elements	
Element	Definition
mime	<i>Value type:</i> string: xs:token <i>Semantic Meaning:</i> The MIME type of this format. <i>Occurrences:</i> required <i>Comments:</i> The format of this string is specified by RFC 2045. The service has to accept this string as a value of the <code>FORMAT</code> parameter.

tr:OutputFormat Metadata Elements	
Element	Definition
alias	<p><i>Value type:</i> string: xs:token</p> <p><i>Semantic Meaning:</i> Other values of FORMAT ("shorthands") that make the service return documents with the MIME type.</p> <p><i>Occurrences:</i> optional; multiple occurrences allowed.</p>
description	<p><i>Value type:</i> string: xs:token</p> <p><i>Semantic Meaning:</i> A human-readable short description of the input format.</p> <p><i>Occurrences:</i> optional</p>

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 VOspace 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, we employ a controlled vocabulary that clients can match against.

The controlled vocabulary is maintained as a StandardKeyEnumeration (see [\[SRE\]](#)) at `ivo://ivoa.org/tap/uploadmethods`. The actual protocols are then fragment identifiers into this document. If necessary, descriptions of the protocols can be obtained from there as well.

At the time of writing, the set of protocol identifiers included:

- **inline** -- HTTP upload as per section 2.5.2 of [\[TAP\]](#).
- **http** -- retrieval from an http URL.
- **https** -- retrieval from an https URL.
- **ftp** -- retrieval from an ftp URL.

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

```
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#http"/>
<uploadMethod ivo-id="ivo://ivoa.org/tap/uploadmethods#ftp"/>
```

Values for the ivo-id attribute which are legal IVORNs but are not members of this controlled vocabulary are not forbidden, but TAP clients cannot in general be expected to understand their semantics.

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 typically have some default value for such a limit. They 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 adjusting the **destructionTime** parameter). Services may put some limit to how far the resource limitations may be raised.

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.

The capabilities element allows the declaration of such limits. These declarations are primarily intended for human consumption and should give guidelines. If a service supports authentication and has different limits depending on what user is authenticated, it should return the limits applying to the logged user.

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

Limits on Time

This document defines two time-like resource limits:

- **retentionPeriod** -- the time from job creation until **destructionTime**; services are free to give the maximum time the destruction time may be set in the future here.
- **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 Definition

```
<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" />
  </sequence>
</complexType>
```

tr:TimeLimits Metadata Elements	
Element	Definition
default	<p><i>Value type:</i> xs:integer</p> <p><i>Semantic Meaning:</i> The value this limit has for newly-created jobs.</p> <p><i>Occurrences:</i> optional</p>
hard	<p><i>Value type:</i> xs:integer</p> <p><i>Semantic Meaning:</i> The value this limit cannot be raised above.</p> <p><i>Occurrences:</i> optional</p>

Limits on Data

Limits on data work like time limits, including having a **default** and a **hard** value, except that both those values have a unit attribute that can either be **bytes** or **rows**.

This document defines two resource limits on data:

- **outputLimit** -- if **unit** is **rows** 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. Since this is not a TAP adjustable

parameter, only `default` is relevant here.

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

tr:DataLimits Type Schema Definition

```
<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" />
  </sequence>
</complexType>
```

tr:DataLimits Metadata Elements	
Element	Definition
default	<p><i>Value type:</i> an integer with optional attributes</p> <p><i>Semantic Meaning:</i> The value this limit has for newly-created jobs.</p> <p><i>Occurrences:</i> optional</p>
hard	<p><i>Value type:</i> an integer with optional attributes</p> <p><i>Semantic Meaning:</i> The value this limit cannot be raised above.</p> <p><i>Occurrences:</i> optional</p>

tr:DataLimit Type Schema Definition

```
<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="bytes" />
            <xs:enumeration value="rows" />
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </simpleContent>
</complexType>
```

tr:DataLimit Attributes	
Attribute	Definition
unit	<p><i>Value type:</i> string with controlled vocabulary</p> <p><i>Semantic Meaning:</i> The unit of the limit specified.</p> <p><i>Occurrences:</i> required</p> <p><i>Allowed Values:</i> bytes, rows</p>

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:

```

<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">
  ...

```

tr:TableAccess Type Schema Definition

```

<xs:complexType name="TableAccess" >
  <xs:complexContent >
    <xs:extension base="tr:TAPCapRestriction" >
      <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" />
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

tr:TableAccess Extension Metadata Elements	
Element	Definition
dataModel	<p><i>Value type:</i> a string with optional attributes</p> <p><i>Semantic Meaning:</i> Identifier of IVOA-approved data model supported by the service.</p> <p><i>Occurrences:</i> optional; multiple occurrences allowed.</p>
language	<p><i>Value type:</i> composite: tr:Language</p> <p><i>Semantic Meaning:</i> The language the service supports.</p> <p><i>Occurrences:</i> required; multiple occurrences allowed.</p>
outputFormat	<p><i>Value type:</i> composite: tr:OutputFormat</p> <p><i>Semantic Meaning:</i> The output format the service supports.</p> <p><i>Occurrences:</i> required; multiple occurrences allowed.</p>
uploadMethod	<p><i>Value type:</i> composite: tr:UploadMethod</p> <p><i>Semantic Meaning:</i> Upload method supported by this service.</p> <p><i>Occurrences:</i> optional; multiple occurrences allowed.</p> <p><i>Comments:</i> The absence of upload methods indicates that the service does not support uploads at all.</p>

tr:TableAccess Extension Metadata Elements	
Element	Definition
retentionPeriod	<p><i>Value type:</i> composite: tr:TimeLimits</p> <p><i>Semantic Meaning:</i> Limits on the time between job creation and DestructionTime</p> <p><i>Occurrences:</i> optional</p>
executionDuration	<p><i>Value type:</i> composite: tr:TimeLimits</p> <p><i>Semantic Meaning:</i> Limits on executionDuration.</p> <p><i>Occurrences:</i> optional</p>
outputLimit	<p><i>Value type:</i> composite: tr:DataLimits</p> <p><i>Semantic Meaning:</i> Limits on the size of data returned.</p> <p><i>Occurrences:</i> optional</p>
uploadLimit	<p><i>Value type:</i> composite: tr:DataLimits</p> <p><i>Semantic Meaning:</i> Limits on the size of uploaded data.</p> <p><i>Occurrences:</i> optional</p>

Appendix A: The Full Schema

```

<?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:tr="http://www.ivoa.net/xml/TAP/v0.1"
version="0.1"
targetNamespace="http://www.ivoa.net/xml/TAP/v0.1"
elementFormDefault="unqualified"
attributeFormDefault="unqualified"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/XMLSchema http://vo.ari.uni-heidelbe
>
  <xs:annotation>
    <xs:appinfo>
      <vm:schemaName>TAPRegExt</vm:schemaName>
      <vm:schemaPrefix>xs</vm:schemaPrefix>
      <vm:targetPrefix>tr</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" schemaLocatior
  <xs:import namespace="http://www.ivoa.net/xml/VODataService/v1.0" schemaLocat
  <xs:complexType name="TAPCapRestriction" abstract="true">
    <xs:annotation>
      <xs:documentation>
        An abstract capability that fixes the standardID to the
        IVOA ID for the TAP standard.
      </xs:documentation>
      <xs:documentation>
        See vr:Capability for documentation on inherited children.
      </xs:documentation>
    </xs:annotation>
  </xs:complexType>

```

```

</xs:annotation>
<xs:complexContent>
  <xs:restriction base="vr:Capability">
    <xs:sequence>
      <xs:element name="validationLevel"
        type="vr:Validation" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="description" type="xs:token"
        minOccurs="0"/>
      <xs:element name="interface" type="vr:Interface"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="standardID" type="vr:IdentifierURI"
      use="required" fixed="ivo://ivoa.net/std/TAP"/>
  </xs:restriction>
</xs:complexContent>
</xs:complexType>

<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="tr:TAPCapRestriction">
      <xs:sequence>

        <xs:element name="dataModel" type="tr: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="tr:Language"
          minOccurs="1" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>
              The language the service supports.
            </xs:documentation>
          </xs:annotation>
        </xs:element>

        <xs:element name="outputFormat" type="tr:OutputFormat"
          minOccurs="1" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>
              The output format the service supports.
            </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 this 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
            DestructionTime
        </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:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="DataModelType">
    <xs:annotation>
        <xs:documentation>
            An IVOA defined data model, identified by an IVORN
            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="vr:IdentifierURI">
      <xs:annotation>
        <xs:documentation>
          The IVORN 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:documentation>
      Each language element corresponds to one version of one
      language. Either name alone or name-version can be
      used as values for the server's LANG parameter. When used
      without a version, the server will use the version of the
      language it deems most appropriate.
    </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="xs:token"
      minOccurs="1" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>
          One version of the language supported by the service.
        </xs:documentation>
        <xs: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>
      </xs:annotation>
    </xs:element>

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

```

```

<xs:element name="userDefinedFunction"
  type="tr:UserDefinedFunction"
  minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>
      Nonstandard functions available on this service.
    </xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="OutputFormat">
  <xs:annotation>
    <xs:documentation>
      An output format supported by the service.
    </xs:documentation>
    <xs:documentation>
      All TAP services must support VOTable output, preserving
      the MIME type of the input. Other output formats are
      optional.

      The primary identifier for an output format is the MIME
      type. If you want to register an output format, you must
      use a MIME type (or make one up using the x- syntax), although
      the concrete MIME syntax is not enforced by the schema.
    </xs:documentation>
  </xs:annotation>

  <xs:sequence>
    <xs:element name="mime" type="xs:token">
      <xs:annotation>
        <xs:documentation>
          The MIME type of this format.
        </xs:documentation>
        <xs:documentation>
          The format of this string is specified by RFC 2045.
          The service has to accept this string as a
          value of the FORMAT parameter.
        </xs:documentation>
      </xs:annotation>
    </xs:element>

    <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 MIME type.
        </xs:documentation>
      </xs:annotation>
    </xs:element>

    <xs:element name="description" type="xs:token" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          A human-readable short description of the
          input format.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>

```

```

</xs:complexType>

<xs:complexType name="UploadMethod">
  <xs:annotation>
    <xs:documentation>
      An upload method as defined by IVOA.
    </xs:documentation>
    <xs:documentation>
      Upload methods are always identified by an IVORN.
      Descriptions can be obtained by dereferencing this
      IVORN.

      A typical service would support

      ivo://ivoa.org/tap/uploadmethods#inline,
      ivo://ivoa.org/tap/uploadmethods#http, and
      ivo://ivoa.org/tap/uploadmethods#https.

      These values come from a StandardKeyEnumeration
      maintained by IVOA. To see the available values,
      ask for the ivo://ivoa.org/tap/uploadmethods resource
      on any registry.

      In principle, you could define IVORNs yourself. However,
      it is recommended that you add your protocol to the
      uploadmethods StandardKeyEnumeration; see the contact
      information on the resource record on how to do that.
    </xs:documentation>
  </xs:annotation>

  <xs:complexContent>
    <xs:restriction base="xs:anyType">
      <xs:attribute name="ivo-id" type="xs:anyURI">
        <!-- We'd like this to be of type vr:IdentifierURI, but that doesn't
        allow fragments. -->
        <xs:annotation>
          <xs:documentation>
            The IVORN of the upload method.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<!-- it would be nice if we could build TimeLimits and DataLimits
in the fashion of java generics - is there a Schema way of doing
this? -->

<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 this limit has for newly-created jobs.

```

```

        </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.
        </xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="DataLimits">
    <xs:annotation>
        <xs:documentation>
            Limits on data sizes, given in rows or bytes.
        </xs:documentation>
    </xs:annotation>

    <xs:sequence>
        <xs:element name="default" type="tr:DataLimit"
            minOccurs="0" maxOccurs="1">
            <xs:annotation>
                <xs:documentation>
                    The value this limit has for newly-created jobs.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="hard" type="tr:DataLimit"
            minOccurs="0" maxOccurs="1">
            <xs:annotation>
                <xs:documentation>
                    The value this limit cannot be raised above.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DataLimit">
    <xs:annotation>
        <xs:documentation>
            A limit on some data size, either in rows or in bytes.
        </xs:documentation>
    </xs:annotation>

    <xs:simpleContent>
        <xs:extension base="xs:integer">
            <xs:attribute name="unit" use="required">
                <xs:annotation>
                    <xs:documentation>
                        The unit of the limit specified.
                    </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="bytes"/>
                        <xs:enumeration value="rows"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>

```

```

    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="UserDefinedFunction">
  <xs:annotation>
    <xs:documentation>
      A description of a nonstandard function available on this
      server.
    </xs:documentation>
    <xs:documentation>
      These descriptions consist of a signature that is in
      principle machine-parseable and a description intended
      for humans.
    </xs:documentation>
  </xs:annotation>

  <xs:sequence>
    <xs:element name="signature" type="xs:token">
      <xs:annotation>
        <xs:documentation>
          The function's signature.
        </xs:documentation>
        <xs:documentation>
          The signature of a function in string form as defined in
          the TAPRegExt recommendation.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="description" type="xs:string"
      minOccurs="0">
      <xs:annotation>
        <xs:documentation>
          Human-readable freeform documentation for the user
          defined function.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

Appendix B: Implementations

At the time of writing, the methods and data structures described in this documents are implemented at and interoperate between:

- [The GAVO Data Center](http://vo.ari.uni-heidelberg.de/soft), in the TAP service. The driving software is available for download at <http://vo.ari.uni-heidelberg.de/soft>.

References

- [\[ADQL\]](#), Pedro Osuna and Inaki Ortiz (eds), 2008: IVOA Astronomical Data Query Language, Version 2.0, IVOA Recommendation
- [FILL-IN](#) ObsTAP
- [\[REGWG\]](#) Various: Registry Working Group home page.
- [\[RFC2045\]](#), N. Freed and N. Borenstein, 1996: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", IETF RFC 2045

- [\[RI\]](#) Kevin Benson, Ray Plante, Elizabeth Auden, Matthew Graham, Gretchen Greene, Marin Hill, Tony Linde, Dave Morris, Wil O'Mullane, Guy Rixon, Kona Andrews, 2008, "IVOA Registry Interfaces v1.0", IVOA Recommendation
- [\[SCHEMA\]](#) IVOA Schema Collection
- [\[SRE\]](#) Paul Harrison (ed), et al, 2010, "StandardsRegExt: a VOResource Schema Extension for Describing IVOA Standards", IVOA Working Draft 19 May 2010
- [\[TAP\]](#) Patrick Dowler, Guy Rixon, Doug Tody, 2010: "Table Access Protocol", IVOA Recommendation 27 March 2010
- [\[UWS\]](#) Paul Harrison, Guy Rixon, 2010: Universal Worker Service Pattern Version 1.0, IVOA Recommendation 10 October 2010
- [\[VOR\]](#) Raymond Plante, Kevin Benson, Matthew Graham, Gretchen Greene, Paul Harrison, Gerard Lemson, Tony Linde, Guy Rixon, Aurélien Stébé, 2008: VOResource: an XML Encoding Schema for Resource Metadata, Version 1.03, IVOA Recommendation 22 February 2008
- [\[VOSI\]](#) Matthew Graham, Ray Plante, Guy Rixon (Eds), 2010: IVOA Support Interfaces Version 1.0, IVOA Proposed Recommendation 06 December 2010