VAMDC XSAMS Consumer Service Specification

Document Information

Editors:	G. Rixon, M. Doronin
Authors:	G. Rixon
Contributors:	VAMDC WP6 working group
Type of document:	standards documentation
Status:	draft
Distribution:	public
Work package:	WP6
Version:	11.12
Date:	21/12/2011
Document code:	
Document URL:	http://www.vamdc.org/documents/xsams-consumer_v11.12.pdf

Abstract: Applications to process data in XSAMS format may be made available as web sites, which makes them accessible for interactive use, or as web services, making them accessible to scripts and other software. This standard prescribes a form for these web applications that includes both the interactive web-site and scriptable web-service.

Version History

Version	Date	Modified By	Description of Change
V11.12	21/12/2011	G. Rixon	first version

Acknowledgements

VAMDC is funded under the "Combination of Collaborative Projects and Coordination and Support Actions" Funding Scheme of The Seventh Framework Program. Call topic: INFRA-2008-1.2.2 Scientific Data Infrastructure. Grant Agreement number: 239108.

CONTENTS

CHAPTER

ONE

UNIFORM PROTOCOL FOR A WEB APPLICATION CONSUMING XSAMS

Applications to process data in XSAMS format may be made available as web sites, which makes them accessible for interactive use, or as web services, making them accessible to scripts and other software. This standard prescribes a form for these web applications that includes both the interactive web-site and scriptable web-service.

Web applications conforming to this standard can be registered in the VAMDC registry. Registration makes the applications available to generic UIs such as the VAMDC portal.

Conforming web applications can read data either from a URL (e.g. the portal passes a data-extract URL leading to a VAMDC database) or from an uploaded file (e.g. a user loads data from a file on his desktop).

1.1 General nature of the web application

The web application consists in a set of web resources (where a "web resource" is anything that has its own URL) arranged in a tree structure. This standard specifies the web resources that must be provided, including their names, semantics and positions in the tree. A web application may also provide other web resources.

All the required web-resources must be available via HTTP v1.1.

None of the required web-resources use the Simple Object Access Protocol (SOAP). All these resources follow the paradigm Representational State Transfer (REST).

The core of the application is a web resource called, in this standard, its primary result. This resource represents the result of the application's processing of one or more XSAMS inputs. The application must have exactly one primary result, implying that all the inputs are combined.

The primary result may be machine-readable (e.g. a transformation of the XSAMS input into HITRAN's legacy format) or human readable (i.e. a web page). If human readable, it may well link to further pages showing related results (e.g. the primary result might be a web page showing a table of atomic states, with links to the details of each state). A machine-readable result may also contain links, but this standard does not define how the links are accessed. Hence, a generic client, such as the VAMDC portal, would not know how to follow the links.

An HTTP request to the primary result specifies the input data, either by passing URLs for those or by including them in the request. Therefore, a client application or script can call the primary result directly as a web service.

To make the application accessible from web browsers, the application must also supply a web page that leads the user to the primary result, allowing him to specify the input data. This resource is typically an HTML form and is referred to as "the form" in the rest of this standard.

Applications following this standard are expected to be registered, so they must provide a "VOSI capabilities" resource to convey registration details to the VAMDC registry.

Some of the registration details are repeated, in human-readable form, on an information page.

The application should also provide a "VOSI availability" resource to allow checking of the system.

1.2 Details of the web resources

1.2.1 The form

The form must be the root resource of the application. It must be available via HTTP GET.

The form must allow the user to specify the input data either by giving URLs (which the user might copy and paste from some other UI) or by uploading files from the desktop. Both modes must be available.

The appearance and behaviour of the form should be fixed. It should not vary according to parameters in the URL that displays it, or in response to information cached in the user's browser session. If the form is made adaptable via parameters or responsive to session history, then it must be useable in the absence of this information.

The author of the application may choose freely the content of the form provided that it meets the requirements above.

1.2.2 The primary result

The primary result is at the URL /service relative to the root URL.

In a given application, the primary result may have any MIME-type, but it must always have the same MIME-type in that application.

The primary result must accept all of the following ways to specify the input data.

- In HTTP GET, via the parameter called `url`, embedded in the URL used to call the application. The value of this parameter is the URL leading to the XSAMS data, and that URL must be in the HTTP scheme. The URL for the input data must be URL encoded before embedding in the URL for the primary result. (This necessary encoding makes it infeasible for a user to simply type the URL in a browser's address-bar, hence the need for the form to invoke the primary result.) The parameter may be repeated to specify more than one XSAMS document.
- In HTTP POST, via the parameter called `url` as above, except that the parameter is written in the body of the request instead of embedding in the URL of the primary result.
- In HTTP POST, via the parameter called `upload`, written in the body of the request. The value of the parameter is the XSAMS document itself, packaged according to internet RFC 1867. This parameter may be repeated to upload multiple, XSAMS documents.

The application is only required to process XSAMS inputs and should typically reject other types.

The application may be written to process a single XSAMS document, a fixed number of documents, any number of documents up to a fixed limit, or any number without limit. If the request contains the wrong number of documents, then the application must reject the request.

When responding to an HTTP request for the primary result, the application may choose whether to cache the result. If the result is not cached, the application must return it directly as the body of the response. If the result is cached, the cached version must be a new web-resource and the application must send an HTTP response redirecting the client to this latter resource.

The HTTP response to a request for the primary result must be one of the following.

- Status code 200,"success", with the result as the body of the response. This applies to uncached results.
- Status code 201, "created", with the `Location` header listing the URL for the cached result. The body of the response must be a web page containing a link to the cached resource (web browsers do not deal with 201 responses automatically). This response indicates that the application has just created a new copy of the result in the cache.
- Status code 303, "see other" with the `Location` header listing the URL for the cached result. No body is needed with this response (web browsers will automatically redirect the user to the stated location).
- Status code 400, "bad request", with a web page explaining the the failure in the body of the response. This code implies that the application operated correctly but the request was inappropriate; e.g. a request

containing the wrong number of inputs; or the wrong type of inputs or a URL for an input that cannot be read. Requests receiving this response should not be repeated by the client.

- Status code 500, “internal server error”, 502, “bad gateway”, 503 “unavailable” or 504, “gateway timeouts”, indicating a problem inside the application. This code indicates that the request was correct but the application failed to process it. Requests receiving this response might be processed correctly at some later date.

1.2.3 VOSI capabilities

The VOSI capabilities are a single XML-document at the URL /capabilities relative to the root resource. A “capability” is an XML fragment describing a particular aspect of an application. The general rules for VOSI capabilities are defined by IVOA’s VOSI standard.

For applications conforming to the current standard, there must be a capability following the schema `http://www.vamdc.org/xml/XSAMS-consumer/v1.0`. Such a capability provides two access URLs, one for the form (of type `WebBrowser`) and one for the primary result (of type `ParamHTTP`).

The following code shows a sample capabilities-document, with the namespaces and locations of schema filled in:

```
<?xml version="1.0" encoding="UTF-8"?>

<cap:capabilities
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:cap="http://www.ivoa.net/xml/VOSICapabilities/v1.0"
  xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.0"
  xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
  xmlns:xc="http://www.vamdc.org/xml/XSAMS-consumer/v1.0"
  xsi:schemaLocation="
  http://www.ivoa.net/xml/VOSICapabilities/v1.0 http://www.vamdc.org/downloads/xml/VOSI-capabilit
  http://www.ivoa.net/xml/XSAMS-consumer/v1.0 http://www.ivoa.net/xml/XSAMS-consumer/v1.0
  http://www.ivoa.net/xml/VOResource/v1.0 http://www.ivoa.net/xml/VOResource/v1.0
  http://www.ivoa.net/xml/VODataService/v1.0 http://www.ivoa.net/xml/VODataService/v1.0">

  <capability standardID="ivo://vamdc/std/XSAMS-consumer" xsi:type="xc:XsamsConsumer">
    <interface xsi:type="vr:WebBrowser">
       <accessURL>http://some.server/some/app</accessURL>
    </interface>
    <interface xsi:type="vs:ParamHTTP">
      <accessURL>http://some.server/some/app/service</accessURL>
      <resultType>text/html</resultType>
    </interface>
    <versionOfStandards>11.12</versionOfStandards>
    <versionOfSoftware>whatever</versionOfSoftware>
    <numberOfInputs>1-100</numberOfInputs>
  </capability>

</cap:capabilities>
```

1.2.4 VOSI availability

The VOSI availability is a single XML-document at the URL /availability relative to the root resource.

The general rules for VOSI availability are defined by IVOA’s VOSI standard.

1.2.5 Service information

The service information is a web page at the URL /info relative to the root resource. It repeats some of the information in the VOSI capabilities in human-readable form.

The page must contain at least the following information.

- Service name
- Description of the work done by the service from a scientific point of view.
- URL for root resource
- Version of this standard supported.
- Number of XSAMS inputs required.
- Whether or not the primary result is cached.
- How long results remain in the cache.

1.3 Caching policy

Caching of the primary result makes it easier to chain together application and makes interactive applications more responsive. The cost of caching is greater complexity and subtlety in the operation of the application.

Applications may cache results implicitly or explicitly. Explicit caching exposed the cached copy as a new web-resource with its own URL. Implicit caching changes the behaviour of the primary result to use the cached copy. The author of an application may choose between implicit caching, explicit caching or no caching at all. The choice must be stated in the registration of the application.

For explicit caching, the application returns status code 201 (or 303) for the request that creates the cached copy. The web resource for those cached results is then immutable. After some lifetime, chosen by the application author and stated in the application registration, the resource is deleted from the web application. Requesting the primary result again, for a given set of inputs, refreshes the cache.

The 201, "created", and 303, "see other" status-codes have essentially the same effect. The only practical difference is that browsers redirect automatically to the indicated web-resource for the 303 code and not for the 201 code. Therefore, the 201 code is better when the application is intended for use from scripts and the 303 code when the application is only used from browsers.

For implicit caching, the application must either maintain the freshness of the results (e.g. using HEAD requests on the URLs for the original data to detect updates), or must supply interactive controls to the user for refreshing the cache. Maintaing cache freshness is hard to implement reliably (and impossible in the case of data uploaded from file), so implicit caching is most applicable to interactive applications where the user can control the refreshing.

Explicitly-cached data are, implicitly, available to any client or user; no access controls are applied. However, the application should not advertise the existence of these data to other users.

1.4 Registration

The application should be registered in the VAMDC registry. This makes it visible to generic UIs such as the VAMDC portal.

If registered, the registration-document type must be `{http://www.ivoa.net/xml/VOResource/v1.0}Service` as defined in the IVOA standard for registration. The registration must include the capability data taken from the VOSI-capabilities resource of the application, as detailed above.

Generic UIs will typically present users with a list of XSAMS-consuming applications. The `title` element of the application's registration-document should be suitable to distinguish the application in such a list: it should state explicitly but tersely what the application does.

1.5 Closely-related applications

There may arise sets of applications with closely related functions; e.g., format converters for different output-formats. There is a natural instinct to combine these in one application where the outputs are distinguished by an extra parameter on the primary result that is specific to that combined application. This approach fails because the generic clients do not understand the special parameter. An application must not rely on custom parameters on the primary result *if the values of those parameters must be chosen by the client*.

Two methods are allowed for combining applications: multiple registrations and onward links from web pages.

Multiple registrations means that the complete set of web resources specified above is replicated for each kind of primary result, but the resources are served by the same web application. Each set of resources is registered separately and appears to clients as a separate application. E.g., for the format-converters, we might provide these resources:

```
converter?format=csv
converter/service?format=csv
converter/capabilities?format=csv
converter?format=lamda
converter/service?format=lambda
converter/capabilities?format=lambda
...
```

These URLs differ only in the parameters, but because they are all registered the clients do not need to choose the parameter values.

Onward links means that the primary result is a web page and contains links to multiple, related results. This approach works only when the application is used interactively.

CHAPTER

TWO

REFERENCES

- International Virtual Observatory Alliance (IVOA) : http://www.ivoa.net
- VOResource: an XML Encoding Schema for Resource Metadata : http://www.ivoa.net/Documents/REC/ReR/VOResource-20080222.html
- Virtual Observatory Support Interfaces (VOSI) : http://www.ivoa.net/Documents/VOSI/20100311
- XSAMS consumer service registration schema : http://www.vamdc.org/xml/XSAMS-consumer/v1.0