Universal Worker Service Pattern - Proposed modifications

IVOA Note 2013 May 13

Author(s):
J. Normand, J-C. Malapert, P. Le Sidaner, A. Schaaff

Abstract

Proposed modifications for the Universal Worker Service (UWS) pattern.

The modifications are about the second part of Universal Worker Service Pattern (v1.0) recommendation (P. Harrison, G. Rixon).

Status of this document

This is a Note. This is the first release of this document.

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.

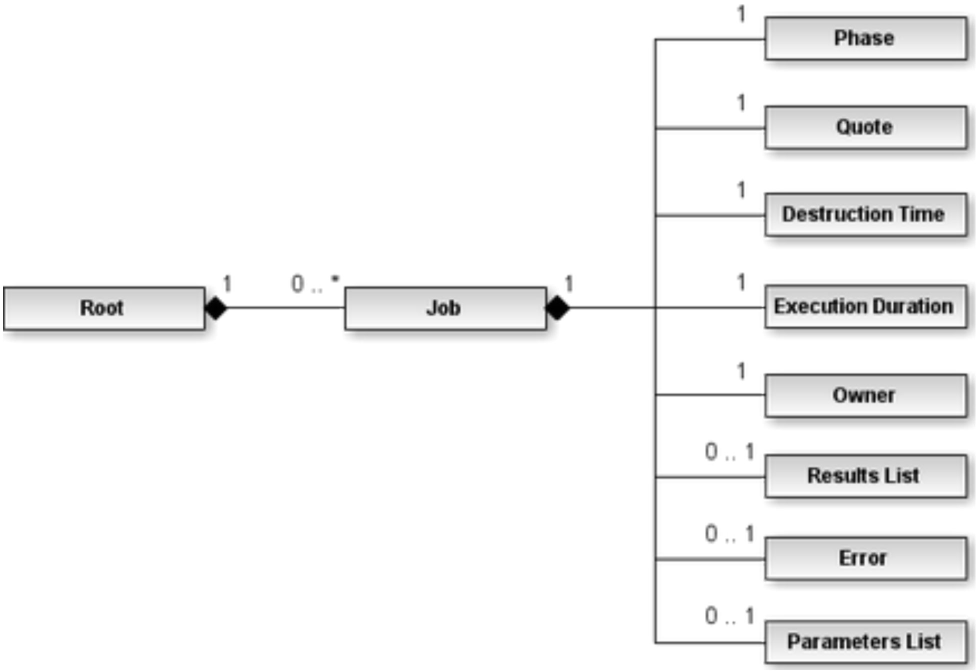
A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Contents

2. Universal Worker Service Interface

2.1 Objects

As illustrated in figure 1, a UWS service consists logically in a set of objects that may be read and written to in order to control jobs.



2.1.1 Jobs

The Jobs object contains all the jobs in the UWS.

2.1.2 Job

The Job object contains all the information relative to one job. Each Job contains:

- Exactly one Phase object
- Exactly one Owner object
- Exactly one Completion time object
- Exactly one Destruction time object
- Zero or one Quote object
- Zero or one Execution duration object
- Zero or one Error object
- Zero or one Results List object
- Zero or one Parameters List object

The v1.1 introduces a new mandatory object `Completion time` which makes optional the `Quote` and `Execution duration` objects.

2.1.3 Phase

The job is treated as a state machine. The Phase object names the state. The values for Phase are:

QUEUED: The job is committed to execution by the client but the service has not yet assigned it to a processor. No results are produced in this phase.

EXECUTING: The job has been assigned to a processor. Results may be produced at any time during this phase.

COMPLETED: The execution of the job is over. Results may be collected at this time.

ERROR: The job failed to complete. Results may be available at this time but they should not be trusted.

ABORTED: The job has been aborted by the system because execution duration has been exceeded.

SUSPENDED: The job has been suspended by the system during execution.

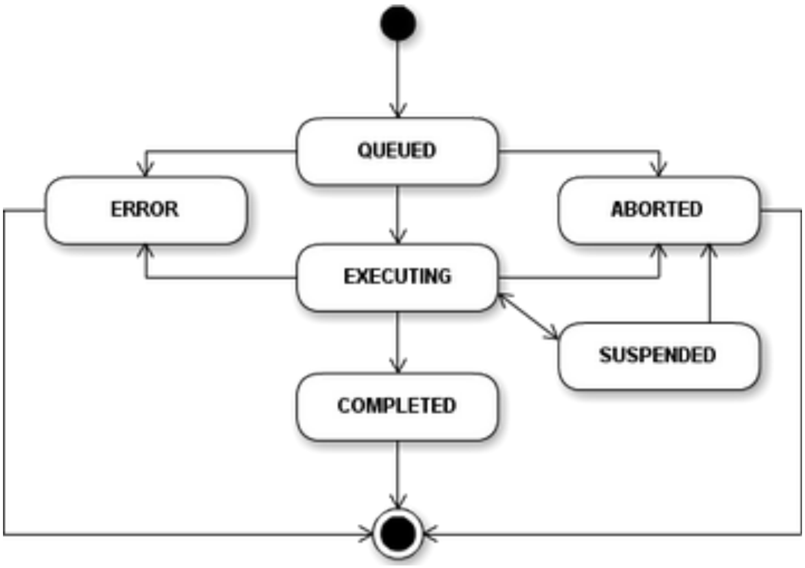
UNKNOWN: The job is in an unknown state.

PENDING: OPTIONAL. This state is used only in the optional calling sequence. The job is accepted by the service but not yet committed for execution.

A successful job will normally progress through the QUEUED, EXECUTING, COMPLETED phases in that order. At any time before the COMPLETED phase a job may be ABORTED, SUSPENDED or may suffer an ERROR. If the UWS reports an UNKNOWN phase, then all the client can do is re-query the phase until a known phase is reported.

When a job has been SUSPENDED, the UWS will automaticaly resume the job into the EXECUTING phase without any intervention.

When a job has been ABORTED, any previously generated results of the job are still available until destruction time.



2.1.4 Quote

The Quote object represents the instant when the job is likely to complete.

2.1.5 Owner

The Owner object represents the identifier of the creator of the job.

When an authentication mechanism is used in the UWS, the implementation should set the Owner object to the identity obtained by the authentication. If there was no authenticated job creator then this should be set to NULL.

2.1.6 Destruction time

The Destruction time object represents the instant when the job shall be deleted.

The initial destruction time is set by the service when the job is created. The client may change the life expectancy of a job but the service may forbid this change or may set limits on the allowed Destruction time. The format is ISO8601.

When the Destruction time has been exceeded, the service automaticly deletes the job and generated results. Any previously generated results of the job are no longer available.

2.1.7 Execution duration

The Execution duration object represents the duration for which a job shall run.

The initial execution duration is set by the service when the job is created. The client may change this duration but the service may forbid this change or may set limits on the allowed Execution duration. The duration is defined in real clock seconds by an integer.

When the Execution duration has been exceeded, the service automaticly aborts the job. Any previously generated results of the job are still available.

2.1.8 Error

The Error object represents a human readable error message for the job.

2.1.9 Results

The Results object represents a list of job Result object.

2.1.10 Result

A job Result is an object resulting from the computation that may be fetched from the service when the job is executing or has completed.

2.1.11 Parameters

The Parameters object represents a list of the job Parameter.

2.1.12 Parameter

The Parameter object represents a parameter of the job.

The parameters have to be set when the job is created.

2.1.13 Completion time

The Completion time object is an absolute time and it respresents the instant when the results will be available.

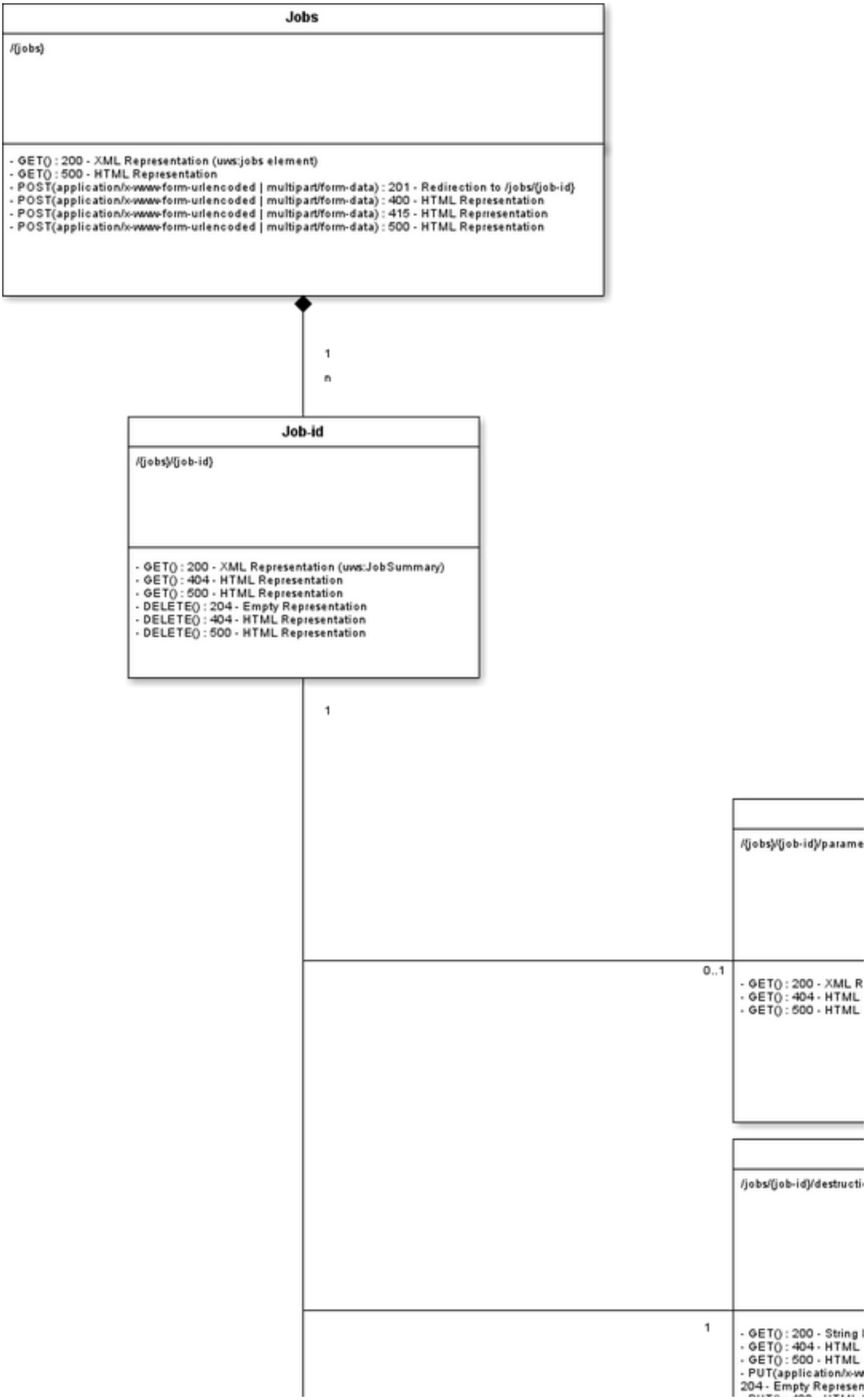
The Completion time (or at least an estimation) is set by the service when the job is created. It can not be changed by the client. The format is ISO8601.

When the Completion time has been reached, the client can access to the generated results.

2.2 Resources

2.2.1 Identification

In a REST binding, the domain model is represented as distinct web-resources each with its own URI.


Jobs
/{jobs}
- GET() : 200 - XML Representation (uws:jobs element)
- GET() : 500 - HTML Representation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 201 - Redirection to /jobs/{job-id}
- POST(application/x-www-form-urlencoded | multipart/form-data) : 400 - HTML Representation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 415 - HTML Representation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 500 - HTML Representation
1
n
Job-id
/{jobs}/{job-id}
- GET() : 200 - XML Representation (uws:JobSummary)
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
- DELETE() : 204 - Empty Representation
- DELETE() : 404 - HTML Representation
- DELETE() : 500 - HTML Representation
1
0..1
/{jobs}/{job-id}/parame
- GET() : 200 - XML R
- GET() : 404 - HTML
- GET() : 500 - HTML
1
/jobs/{job-id}/destructi
- GET() : 200 - String
- GET() : 404 - HTML
- GET() : 500 - HTML
- PUT(application/x-w
204 - Empty Represen

In the REST interface of UWS, each objects defined above is available as a web resource with its own URI. The URIs must form a hierarchy as shown in the table below:

URI	Object
`/{jobs}`	Job List
`/{jobs}/{job-id}`	Job
`/{jobs}/{job-id}/phase`	Phase of job {job-id}
`/{jobs}/{job-id}/quote`	Quote of job {job-id}
`/{jobs}/{job-id}/owner`	Owner of job {job-id}
`/{jobs}/{job-id}/destruction`	Destruction instant of job {job-id}
`/{jobs}/{job-id}/completiontime`	Completion time of job {job-id}
`/{jobs}/{job-id}/executionduration`	Execution duration of job {job-id}
`/{jobs}/{job-id}/error`	Error message of job {job-id}
`/{jobs}/{job-id}/results`	Results List of job {job-id}
`/{jobs}/{job-id}/results/{result-id}`	The Result {result-id} of job {job-id}
`/{jobs}/{job-id}/parameters`	Parameters List of job {job-id}
`/{jobs}/{job-id}/parameters /{parameter-name}`	The parameter {parameter-name} of job {job-id}

The service implementer is free to choose the names given in parentheses above. The other names are part of the UWS standard.

The URI for the Job List, in its absolute form is the root URI for the whole UWS. This URI should be given as the access URL in the UWS registration.

2.2.2 Representation

Each of the UWS objects is mapped to a resource with its own URI as detailed in the table above, and for each URI, a HTTP GET fetches a representation of that resource.

In UWS 3 types of representations are used depending on objects and context: XML, plain text and HTML.

If an object is a container for other objects (Job List, Job, Result List, Parameter List) then an XML representation of the object should be returned, otherwise for simple atomic types (Completion time, Quote, Execution Duration...) a plain text representation (mimetype: "text/plain") should be returned.

HTTP allows multiple representations of a resource distinguished by their MIME types and selected by the HTTP "Accept" headers of a HTTP GET request. A UWS implementation can exploit this to support both web browsers and rich clients in the same tree of resources. Although the default behaviour is to return XML, a UWS could return HTML or XHTML to clients that accept these types. These clients are assumed to be web browsers and the UWS is generating its own user interface. The HTML interface generated should allow full control of the UWS via the use of HTML forms and appropriate links.

Clients which are assumed to be part of remote applications that drive UWS without showing the details to their users should accept only "application/xml,text/plain" type. A UWS must therefore return XML representations of the resources in preference to the HTML representation. A technique that may be used to always return XML that modern browsers can transform on the client-side to HTML is via the xml-stylesheet processing instruction, which can be used to point to a suitable XSL resource to perform the transformation.

2.2.2.1 Job List

For the Job List object an XML representation must be returned. This representation is specified by the `<uws:jobs>` element in the UWS schema.

The representation of the Job List is a list of links to the resources representing the jobs. The list may be empty if the UWS is idle.

```
<?xml version="1.0" encoding="UTF-8"?>
<uws:jobs xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 http://ivoa.net/xml/UWS/UWS-v
      xmlns:xlink="http://www.w3.org/1999/xlink">
   <uws:jobref id="job1" xlink:href="http://uws.example.org/jobs/job1">
      <uws:phase>COMPLETED</uws:phase>
   </uws:jobref>
   <uws:jobref id="job2" xlink:href="http://uws.example.org/jobs/job2">
      <uws:phase>EXECUTING</uws:phase>
   </uws:jobref>
</uws:jobs>
```

2.2.2.2 Job

For the Job object an XML representation must be returned. This representation is specified by the `<uws:job>` element in the UWS schema.

The `<uws:job>` element has placeholders of all of the standard UWS objects, and in addition there is a `<uws:jobinfo>` element which can be used by implementations to include any extra information within the job description. An example of such a job instance is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<uws:job xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 http://ivoa.net/xml/UWS/UW
         xmlns:xlink="http://www.w3.org/1999/xlink">
   <uws:jobId>a108c361-d140-f724-8936-1b1cff656f05</uws:jobId>
   <uws:phase>COMPLETED</uws:phase>
   <uws:startTime>2011-09-07T09:02:19+00:00</uws:startTime>
   <uws:endTime>2011-09-07T09:02:52+00:00</uws:endTime>
   <uws:executionDuration>86400</uws:executionDuration>
   <uws:destruction>2011-09-14T09:02:12+00:00</uws:destruction>
   <uws:completionTime>2011-09-13T09:02:12+00:00</uws:completionTime>
   <uws:ownerId xsi:nil="true"/>
   <uws:quote xsi:nil="true"/>
   <uws:parameters>
      <uws:parameter id="param1">value1</uws:parameter>
      <uws:parameter id="param2" byReference="true">http://uws.example.org/jobs/job1/par
   </uws:parameters>
   <uws:results>
      <uws:result id="result1" xlink:href="http://uws.example.org/jobs/job1/results/resu
   </uws:results>
</uws:job>
```

2.2.2.3 Results List

For the Results List object an XML representation must be returned. This representation is specified by the `<uws:results>` element in the UWS schema.

The representation of the Results List is a list of links to the resources representing the results. These resources may have any URI and any MIME type. A sensible default for their URIs is to make them children of `/{jobs}/{job-id}/results`, but this is not required. It may sometimes be easier for a service implementer to point to a resource on some web server separate from that running the UWS. Therefore, a client must always parse the Results List to find the results. Each result in a result list must be given a unique identifier. Where a protocol applying UWS specifies standard results it must do so fixing the identifier for those results and fixing the result URIs, however the UWS must still return a valid Results List at `/{jobs}/{job-id}/results`, even though in this case the identifiers and URIs could be precomputed by the client.

```
<?xml version="1.0" encoding="UTF-8"?>
<uws:results xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 http://ivoa.net/xml/UWS/UW
         xmlns:xlink="http://www.w3.org/1999/xlink">
   <uws:result id="result1" xlink:href="http://uws.example.org/jobs/job1/results/result1
   <uws:result id="result2" xlink:href="http://uws.example.org/jobs/job1/results/result2
</uws:results>
```

2.2.2.4 Parameters List and parameter

For the Parameters List object an XML representation must be returned. This representation is specified by the `<uws:parameters>` element in the UWS schema.

The representation of the Parameters List is a list of `<uws:parameter>` elements. The form that the parameters take will depend on the JDL of the implementing service.

For services where the JDL consists of a list of name/value pairs (typical of the standard IVOA "simple" access protocols), then these would naturally be expressed in the parameter list. Each of these elements can either represent the value of the parameter directly, where the content of the element is a textual representation of the parameter, or in the case where the parameter value cannot be represented legally within XML (e.g. the parameter is a binary type such as a FITS file) then the content of the element is a URL to the parameter value. To indicate this case the attribute `byReference` is set to `true`.

```
<?xml version="1.0" encoding="UTF-8"?>
<uws:parameters xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 http://ivoa.net/xml/UWS/UWS-v
   <uws:parameter id="name1">value1</uws:parameter>
   <uws:parameter id="name2" byReference="true">http://uws.example.org/jobs/job1/paramet
</uws:parameters>
```

For services where the JDL consists in a document with its own syntax (for instance an XML document with a specific schema, JSON file...), then there would be a single `<uws:parameter>` element where the content was the URL to that document. In this case the attribute `byReference` is set to `true`.

```
<?xml version="1.0" encoding="UTF-8"?>
<uws:parameters xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 http://ivoa.net/xml/UWS/UWS-v
   <uws:parameter id="jdl" byReference="true">http://uws.example.org/jobs/job1/parameter
</uws:parameters>
```

When the value is expressed by reference, the service can give the possibility to upload your document. In this case the URL is given by UWS itself and takes the following form : `http://uws.example.org/{jobs}/{job-id}/parameters/{parameter-name}`.

2.2.2.5 Phase, Completion time, Quote, Owner, Destruction time, Execution duration, Error

For these objects a textual representation must be returned.

2.2.3 Manipulation

2.2.3.1 Job List

To retrieve the job list use HTTP GET method on `/{jobs}` URI. The expected response can be:

200 "OK" - uws:jobs representation
    If succeed.
500 "Internal Server Error" - HTML representation
    If failure.

2.2.3.2 Job

To retrieve the job summary use HTTP GET method on `/{jobs}/{job-id}` URI. The expected response can be:

200 "OK" - uws:jobsummary representation
    If succeed.
404 "Not Found" - HTML representation
    If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
    If failure.

2.2.3.2.1 CREATE

To create a new job (unless the service rejects the request) use HTTP POST method on `/{jobs}` URI. The expected response can be:

201 "Created" - Empty representation
    Redirect to the newly created `/{jobs}/{job-id}` resource if succeed.
400 "Bad Request" - HTML Representation
    If at least one of the POSTed parameter is not a service parameter.
415 "Unsupported Media Type" - HTML Representation
    If the entity of the request is in a format not supported. (see below)
500 "Internal Server Error" - HTML representation
    If failure.

If the value of the parameter can be expressed with a textual representation, the `Content-Type` header of the request should be set to `application/x-www-form-urlencoded`.

For parameters defined by a document, you can either set them by reference or by uploading the document on the UWS.

- To set them by reference, the `Content-Type` header of the request should be set to `application/x-www-form-urlencoded`.
- To upload the document on the UWS, the `Content-Type` header of the request should be set to `multipart/form-data`.

If you set several parameters at the same time including an upload, use `multipart/form-data`.

2.2.3.2.2 DELETE

To delete a job use HTTP DELETE method on `/{jobs}/{job-id}` URI. The expected response can be:

204 "No Content" - Empty representation
 If succeed.
404 "Not Found" - HTML representation
 If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
 If failure.

Some server and client do not handle HTTP DELETE method. In this case we propose to use the tunneling method. The tunneling can use query parameters. This is particularly useful for browser-based applications that can't fully control the HTTP requests sent. In our case, we propose to use METHOD argument to pass the method HTTP in the query.

A job can be deleted at any time.

2.2.3.3 Job parameters

To retrieve all the parameters of a job use HTTP GET method on `/{jobs}/{job-id}/parameters` URI. The expected response can be:

200 "OK" - uws:parameters representation
 If succeed.
404 "Not Found" - HTML representation
 If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
 If failure.

To retrieve a particular parameter of a job use HTTP GET method on `/{jobs}/{job-id}/parameters/{parameter-name}` URI. The expected response can be:

200 "OK" - {parameter-name} representation
 If succeed.
404 "Not Found" - HTML representation
 If {job-id} or {parameter-name} does not exist.
500 "Internal Server Error" - HTML representation
 If failure.

2.2.3.4 Job properties

2.2.3.4.1 DESTRUCTION

To retrieve the destruction time of a job use HTTP GET method on /{jobs}/{job-id}/destruction URI. The expected response can be:

200 "OK" - String representation
 If succeed.
404 "Not Found" - HTML representation
 If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
 If failure.

The initial destruction time is set by the service when the job is created. To change the life expectancy of a job use HTTP PUT method on `/{jobs}/{job-id}/destruction` URI. The service may forbids this change. The expected response can be:

204 "No Content" - Empty representation
If succeed (ie. the destruction time has been changed).
400 "Bad Request" - HTML Representation
If format is not ISO8601.
404 "Not Found" - HTML representation
If {job-id} does not exist.
405 "Method Not Allowed" - HTML representation
If the service forbids this change.
500 "Internal Server Error" - HTML representation
If failure.

2.2.3.4.2 EXECUTIONDURATION

To retrieve the execution duration of a job use HTTP GET method on /{jobs}/{job-id}/executionduration URI. The expected response can be:

200 "OK" - String representation
If succeed.
404 "Not Found" - HTML representation
If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
If failure.

The initial execution duration is set by the service when the job is created. To change this duration use HTTP PUT method on `/{jobs}/{job-id}/executionduration` URI. The service may forbids this change. The expected response can be:

204 "No Content" - Empty representation
If succeed (ie. the execution duration has been changed).
400 "Bad Request" - HTML Representation
If format is not an integer.
404 "Not Found" - HTML representation
If {job-id} does not exist.
405 "Method Not Allowed" - HTML representation
If the service forbids this change.
500 "Internal Server Error" - HTML representation
If failure.

2.2.3.4.3 RESULTS

To retrieve the results of a job use HTTP GET method on /{jobs}/{job-id}/results URI. The expected response can be:

200 "OK" - uws:results representation
If succeed.
404 "Not Found" - HTML representation
If {job-id} does not exist.
500 "Internal Server Error" - HTML representation
If failure.

To retrieve a particular result of a job use HTTP GET method on `/{jobs}/{job-id}/results/{result-id}` URI. The expected response can be:

200 "OK" - {result-id} representation
If succeed.

404 "Not Found" - HTML representation
: If {job-id} or {result-id} does not exist.

500 "Internal Server Error" - HTML representation
: If failure.

2.2.3.4.4 COMPLETION TIME, OWNER, QUOTE, PHASE, ERROR

To retrieve one of these properties of a job use HTTP GET method on /{jobs}/{job-id}/{property} URI (with {property} stands for owner, quote, phase, error or completiontime). The expected response can be:

200 "OK" - String representation
: If succeed.

404 "Not Found" - HTML representation
: If {job-id} does not exist.

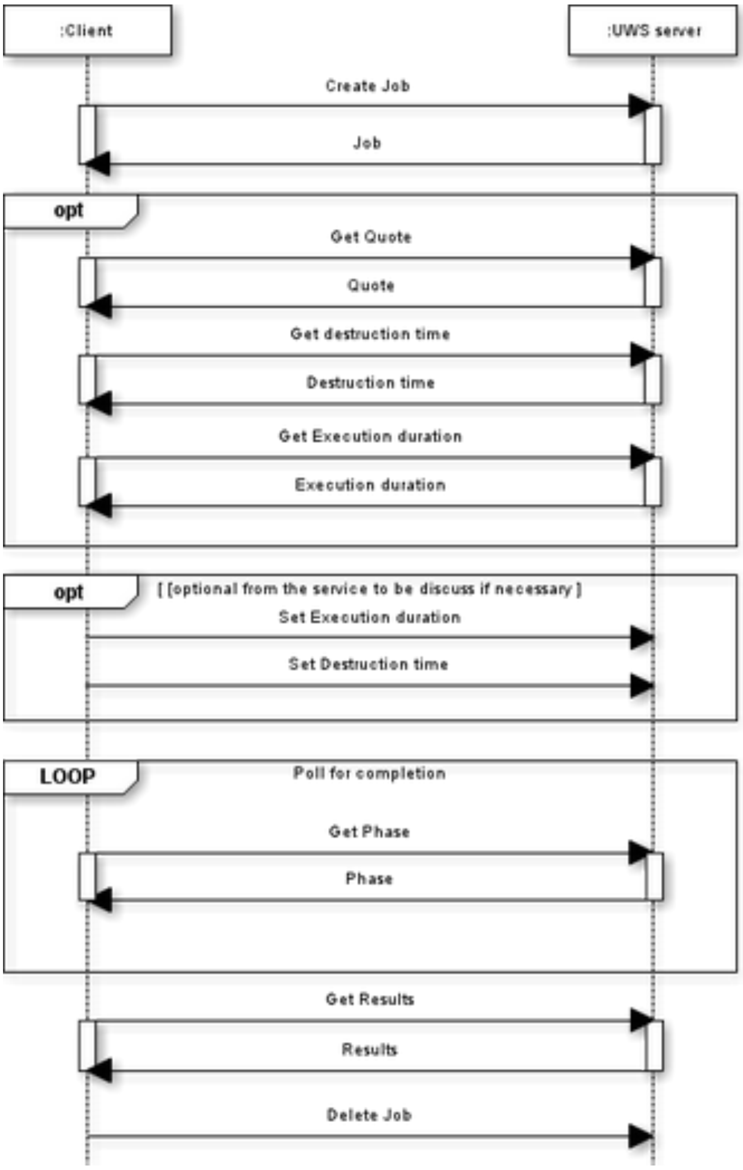
500 "Internal Server Error" - HTML representation
: If failure.

2.3 Calling sequences

There are two ways to start a job:

Mandatory
: The job will progress through the QUEUED, EXECUTING, COMPLETED phases in that order.

Optional
: This is the v1.0 way. The job will progress through the PENDING, QUEUED, EXECUTING, COMPLETED phases in that order. It allows to set parameters after the job creation.



2.4 Service description

It seems necessary to provide to users of an UWS service a description to know how to use it. This description should explain the purpose of the service, the input parameters, the results and the way to use it.

We propose to provide this description with WADL. All the requests (inc. input parameters and results) between the client and the server can be described using WADL.

WADL description also allows to easyly build user interface.

See Appendix A.1 for an example of a service described with WADL.

Changes explanations

This version introduces the following changes. They have been discussed during UWS session in Pune InterOp by P. Le Sidaner.

Optional PENDING phase

The modifications above propose to create, set parameters and start the job in one step. So the first value for Phase will be QUEUED, EXECUTING or ERROR. This scenario is mandatory.

For compatibility reason with the previous version of the standard the scenario which allows to set parameters after the job creation becomes optional.

Aborting a job action becomes optional

This action has been kept for compatibility reason.

Completion time, Quote, Execution duration and Destruction time

Two things are useful for the user:

1. At what time the results will be available (ie. the job is completed).
2. How long time the results will be available.

As users are only interested in completion time, we propose to merge Execution duration and Quote object in a more explicit one: Completion time

The Completion time is an absolute time and it respresents the instant when the results will be available. The service must provide this time or at least an estimation.

Also it is not necessary to provide to the client the ability to change these time.

To be discussed

How to define optional capabilities

Authentication mechanism

Maybe we can use a similar mechanism than http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2

Pagination

Because JobList, ParametersList and ResultsList objects can contain lot of children, we propose to include a pagination mechanism to limit the number of children when these objects are requested. For example:

HTTP GET on `/{jobs}?start=<number of the first job>&extend=<amount of job to retreive>`

Using this mechanism, user have to known the amount of job in the JobList. User can retreive this information using the HTTP HEAD method on `/{jobs}` resource. It will be added as metadata in the header of the response. For example: `header('JobAmount: N')`.

By default HTTP GET on `/{jobs}` returns the first page.

Appendix A.1 WADL description

```
<application xmlns="http://wadl.dev.java.net/2009/02"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xsi:schemaLocation="http://wadl.dev.java.net/2009/02 http://www.w3.org/Submission/wadl/wadl.xs
   xmlns:uws="xmlns:uws=http://www.ivoa.net/xml/UWS/v1.0"
   xmlns:xlink="xmlns:xlink=http://www.w3.org/1999/xlink">
   <doc>Implements the UWS service</doc>
   <grammars>
      <include href="http://ivoa.net/xml/UWS/UWS-v1.0.xsd"/>
   </grammars>
```

```
<!-- TODO: base value -->
<resources base="http://voparis-uws.obspm.fr/uws-v1.0/pdstotiff">
   <resource path="{job-id}">
      <doc>This resource handles a job</doc>
      <method name="DELETE">
         <doc>Delete a job</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="303">
            <doc>Redirects to /{job}</doc>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="GET">
         <doc>Get a job summary</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/xml" element="uws:job"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>Control a Job</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <!-- TODO: parameters -->
            <param style="query" name="pds" type="xs:anyURI" required="true">
               <doc>PDS file (IMG extension)</doc>
            </param>
            <param style="query" name="geo" type="xs:anyURI" required="true">
               <doc>PDS file (GEO extension)</doc>
            </param>
            <!-- END: parameters -->
            <param style="query" name="ACTION" type="xs:string" required="false">
               <option value="DELETE"/>
            </param>
            <representation mediaType="multipart/form-data"/>
            <representation mediaType="application/x-www-form-urlencoded"/>
         </request>
         <response status="303">
            <doc>Redirects to /{job}/{job-id}</doc>
         </response>
         <response status="404"/>
         <response status="415"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/phase">
      <doc>This resource handles job phase</doc>
      <method name="GET">
         <doc>Get job phase</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/plain" element="xs:string"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>Control a Job</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <param style="query" name="PHASE" type="xs:string" required="false">
               <option value="ABORT"/>
               <option value="RUN"/>
            </param>
            <representation mediaType="application/x-www-form-urlencoded"/>
         </request>
         <response status="303">
            <doc>Redirects to /{job}/{job-id}</doc>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/executionduration">
      <doc>This resource handles execution duration</doc>
      <method name="GET">
         <doc>Get job execution duration</doc>
         <request>
            <param style="template" name="job-id" type="xs:int" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/plain" element="xs:int"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>Set execution duration</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <param style="query" name="EXECUTIONDURATION" type="xs:int" required="false"/>
         </request>
         <response status="303">
            <doc>Redirects to /{job}/{job-id}</doc>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/destruction">
      <doc>This resource handles the destruction time of the job</doc>
      <method name="GET">
         <doc>Get job destruction time as ISO8601 format</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/plain" element="xs:dateTime"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>Set job destruction time</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <param style="query" name="DESTRUCTION" type="xs:dateTime" required="false"/>
         </request>
         <response status="303">
            <doc>Redirects to /{job}/{job-id}</doc>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/error">
      <doc>This resource handles error happening during the job processing</doc>
      <method name="GET">
         <doc>Get the generated error</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/xml" element="uws:errorSummary"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/quote">
      <doc title="Quote Resource">This resource handles quote prediction</doc>
      <method name="GET">
         <doc>Get quote value</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/plain" element="xs:dateTime"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/results">
      <doc title="Results Resource">This resource handles job results</doc>
      <method name="GET">
         <doc>Get job results</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/xml" element="uws:results"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/results/{result-id}">
      <doc>This resource handles result value</doc>
      <method name="GET">
         <doc>Get a result</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <param style="template" name="result-id" type="xs:string" required="true">
               <!-- TODO: results name -->
               <option value="0" mediaType="image/tiff"/>
               <!-- END: results name -->
            </param>
         </request>
         <response status="200"/>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/parameters">
      <doc title="Parameters Resource">This resource handles job parameters</doc>
      <method name="GET">
         <doc>Get Job parameters</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/xml" element="uws:parameters"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>Set parameters</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <!-- TODO: parameters -->
            <param style="query" name="pds" type="xs:anyURI" required="true">
               <doc>PDS file (IMG extension)</doc>
            </param>
            <param style="query" name="geo" type="xs:anyURI" required="true">
               <doc>PDS file (GEO extension)</doc>
            </param>
            <!-- END: parameters -->
            <representation mediaType="multipart/form-data"/>
            <representation mediaType="application/x-www-form-urlencoded"/>
         </request>
         <response status="303">
            <doc>Redirect to /{job}/{job-id}</doc>
         </response>
         <response status="404"/>
         <response status="415"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/parameters/{parameter-name}">
      <doc>This resource handles parameter value</doc>
      <method name="GET">
         <doc>Get a parameter name</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
            <param style="template" name="parameter-name" type="xs:string" required="true">
               <!-- TODO: parameters name -->
               <option value="pds" mediaType="image/pds"/>
               <option value="geo" mediaType="text/plain"/>
               <!-- END: parameters name -->
            </param>
         </request>
         <response status="200"/>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource path="{job-id}/owner">
      <doc>This resource handles job owner</doc>
      <method name="GET">
         <doc>Getting an ownerId</doc>
         <request>
            <param style="template" name="job-id" type="xs:string" required="true"/>
         </request>
         <response status="200">
            <representation mediaType="text/plain" element="xs:string"/>
         </response>
         <response status="404"/>
         <response status="500"/>
      </method>
   </resource>
   <resource>
      <doc>This resource contains the whole list of job</doc>
      <method name="GET">
         <doc>List all created jobs. The list may be empty if the UWS is idle</doc>
         <response status="200">
            <representation mediaType="text/xml" element="uws:jobs"/>
         </response>
         <response status="500"/>
      </method>
      <method name="POST">
         <doc>POSTing a request to the Job List creates a new job</doc>
         <request>
            <!-- TODO: parameters -->
            <param style="query" name="pds" type="xs:anyURI" required="true">
               <doc>PDS file (IMG extension)</doc>
            </param>
            <param style="query" name="geo" type="xs:anyURI" required="true">
               <doc>PDS file (GEO extension)</doc>
            </param>
            <!-- END: parameters -->
            <param style="query" name="PHASE" type="xs:string" required="false" fixed="RUN"/>
            <representation mediaType="multipart/form-data"/>
            <representation mediaType="application/x-www-form-urlencoded"/>
         </request>
         <response status="303">
            <doc>Redirects to newly created resource /{job-id}</doc>
         </response>
         <response status="415"/>
         <response status="500"/>
      </method>
   </resource>
```

```
        </resources>
    </application>
```

Jobs
/{jobs}
- GET() : 200 - XML Representation (uws:jobs element)
- GET() : 500 - HTML Representation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 201 - Redirection to /jobs/{job-id}
- POST(application/x-www-form-urlencoded | multipart/form-data) : 400 - HTML Representation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 415 - HTML Reprresentation
- POST(application/x-www-form-urlencoded | multipart/form-data) : 500 - HTML Representation
1
n
Job-id
/{jobs}/{job-id}
- GET() : 200 - XML Representation (uws:JobSummary)
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
- DELETE() : 204 - Empty Representation
- DELETE() : 404 - HTML Representation
- DELETE() : 500 - HTML Representation
1
0..1
Parameters
/{jobs}/{job-id}/parameters
- GET() : 200 - XML Representation (uws:parameters element)
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
1..*
Parameter-name
/{jobs}/{job-id}/parameters/{parameter-name}
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
1
Destruction time
/jobs/{job-id}/destruction
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
- PUT(application/x-www-form-urlencoded [DESTRUCTION=<ISO8601>]) : 204 - Empty Representation
- PUT() : 400 - HTML Representation
- PUT() : 404 - HTML Representation
- PUT() : 405 : HTML Representation
- PUT() : 500 - HTML Representation
1
Execution Duration
/jobs/{job-id}/executionduration
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
- PUT() : 204 - Empty Representation
- PUT() : 400 - HTML Representation
- PUT() : 404 - HTML Representation
- PUT() : 405 : HTML Representation
- PUT() : 500 - HTML Representation
1
Owner
/jobs/{job-id}/owner
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
1
Quote
/jobs/{job-id}/quote
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
1
Phase
/jobs/{job-id}/phase
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
0..1
Results
/jobs/{job-id}/results
- GET() : 200 - XML Representation (uws:Results)
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
1..*
Result
/jobs/{job-id}/results/{resultId}
- GET() : 200 - XML Representation (uws:Results)
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation
0..1
Error
/jobs/{job-id}/error
- GET() : 200 - String Representation
- GET() : 404 - HTML Representation
- GET() : 500 - HTML Representation