

## 1. - Preliminaries

Throughout this document, we will assume that the reader is already familiar with the basic concepts of BioMoby. For more information about BioMoby, please refer to the official documentation.<sup>i</sup>

This document contains the specification of how to deal with asynchronous 'POST' services in BioMoby.

The aim of the specification is to contribute to the standardisation of asynchronous 'POST' service calls in BioMoby. A new extended set of service calling operations and defined XML messages are detailed.

The main motivation for the specification is to facilitate the implementation of "long-running" services, i.e. services that demand enough computational resources to need more than a few minutes to compute the result.

*Note: this document is heavily influenced by the document created by Enrique de Andres, Jose Maria Fernandez Gonzalez, Johan Karlsson, Sergio Ramirez, Jose Manuel Rodriguez Carrasco, Oswaldo Trelles and David Gonzalez-Pisano to describe asynchronous SOAP services in MOBY.*

## 2. - Current BioMoby Specification for POST Services

A MOBY compliant POST service (registered as having the service category 'cgi') is one that uses only object/service classes defined in the MOBY Central registry, presents its service interface via HTTP POST, and registers this service interface in MOBY Central. These services were introduced to allow the SOAPless interaction of BioMoby services and to allow services to produce or consume large amounts of data.

An HTTP POST service can be called through a single procedure (operation) by using the service URL, as registered with the MOBY Central registry, and POSTing, either as raw data or with the parameter 'data' being the full XML input message<sup>ii</sup>. The result of calling an HTTP POST service is a full XML BioMoby output message.<sup>iii</sup>

Several problems related to the long-running services have been identified in BioMoby. The same transaction is used to request the execution of a service and to wait for the results. This causes the system to remain occupied and non-responsive in the meanwhile. Also, it is a *de facto* standard to set a connection timeout both in the server and any clients, closing the sockets being used for client/service communication. Often the connection timeout is no more than a few minutes, making it impossible to invoke long running services.

Recently, a specification and implementation for asynchronous services was added to BioMoby<sup>iv</sup>. This solves the problem above for long running SOAP based services, but not for HTTP POST services. Additionally, since most SOAP implementations parse the SOAP messages using DOM (in memory) based parsers, running out of memory is very common while working with large data sets.

### 3. – Specification of asynchronous services/clients

To enable calls to long-running services, it is best to use an asynchronous communication model. Asynchronous HTTP POST services will utilize WSRF (Web Services Resource Framework), an OASIS standard, in a manner similar to asynchronous SOAP based BioMoby services<sup>v</sup>.

#### Asynchronous HTTP POST services must implement the following WSRF operations

- **GetResourceProperty:** this mandatory WSRF operation allows a requestor to retrieve the value of a single resource property of a WS-RESOURCE.
- **GetMultipleResourceProperties:** this optional WSRF operation allows a requestor to retrieve the values of a multiple resource properties of a WS-RESOURCE.
- **Destroy:** this optional WSRF operation allows a service requestor to request the immediate destruction of a WS-RESOURCE.

Furthermore, both BioMoby asynchronous and synchronous clients must be able to process the following standard WSRF faults:

- **ResourceUnknownFault:** used to indicate that the resource identified in the message is not known to the web service.
- **ResourceUnavailableFault:** used to indicate that the web service is active, but unable to provide access to the resource.
- **InvalidResourcePropertyQNameFault:** used to indicate that resource property specified in the request message did not correspond to a resource property element of the WS-Resource referred to in the request message.
- **ResourceNotDestroyedFault:** used to indicate that a WS-Resource was not destroyed for some reason.

The following groups of properties are mandatory for a BioMoby HTTP POST asynchronous service:

- **Status properties:**
  - This group represents the status of the jobs resulting from a batch-call. The batch-call itself is identified through the EPR, so the name of the property will be used to identify the job. The service must have a property whose QName is “mobyws:status\_queryID” for each job (input mobyData). For example, if a batch-call contains two jobs with queryIDs q1 and q2, then these properties must exist: “mobyws:status\_q1” and “mobyws:status\_q2”. The status will be given using the OMG’s LSAE standard<sup>vi</sup> schema for notification events.
- **Result properties:**
  - This group represents the result from the particular jobs belonging to a batch-call. If a client asks for a result property before a particular job has finished, or the property is ill-formed (i.e. not a valid QName), a WSRF Fault “InvalidResourcePropertyQNameFault” will be returned. The batch-call itself is identified through the EPR, so the name of the property will be used to identify the job. The result properties are QNames similar to the status ones, but following the pattern “mobyws:result\_queryID”.

These properties are read-only. WSRF specifies some optional operations that can be used to delete or modify such properties. Any attempt to change the properties discussed in this specification must be stopped and a WSRF Fault should be returned by the service (“UnableToModifyResourcePropertyFault”, signalling that these properties are read-only).

## Registering HTTP POST Asynchronous Services

Service must be able to indicate to clients if their services can work in an asynchronous mode or not. Such information must be made part of the service registration procedure. The category field of a service instance will be used for storing this information.

Unlike the asynchronous SOAP counterparts, *it is not mandatory to provide a synchronous version of the service*. The reason is that HTTP POST services have yet to gain traction in the BioMoby community, so there is no need to provide backwards compatibility.

An Asynchronous HTTP POST service (category is *cgi-async*), must implement the following HTTP POST operations:

- For submitting a job:
  - A POST to the service’s URL the full BioMoby input XML message
- For polling the status of a job/jobs
  - A POST to the service’s URL with a *’/status’* appended to the URL
  - HTTP header “biomoby-wsrf” set with *GetResourceProperty/GetMultipleResourceProperties* for the property “status\_qname”, where qname is the queryID of the corresponding mobyData input.  
*Note: no new lines are allowed in this header!*
- For retrieving results of a job/jobs
  - A POST to the service’s URL with a *’/results’* appended to the URL
  - HTTP header “biomoby-wsrf” set with *GetResourceProperty/GetMultipleResourceProperties* for the property “result\_qname”, where qname is the queryID of the corresponding mobyData input.  
*Note: no new lines are allowed in this header!*
- For destroying a job/jobs
  - A POST to the service’s URL with a *’/destroy’* appended to the URL
  - HTTP header “biomoby-wsrf” set with a message containing the EPR to the Destroy WSRF operation.  
*Note: no new lines are allowed in this header!*

Please note that although the service provider must implement and publish these additional operations, only the submission operation will be registered with a Moby Central registry. Additionally, Moby Central will produce WSDL for the additional operations listed above.

## Invoking HTTP POST Asynchronous Services – Communication sequence

The sequence for executing an HTTP POST asynchronous service is as follows:

### *The client starts the session*

- The client POSTs the request message to the service URL either as raw data or with the parameter 'data' being the standard MOBY content and data needed to execute the service.

The asynchronous service accepts the task and returns back a WSRF message (containing a batch-call identifier or "ticket" inside of an EPR). From this point, both client and server will work in asynchronous modes.

### *Polling for service execution status*

The clients use the GetResourceProperty/GetMultipleResourceProperties WSRF operation to retrieve status information using the batch-call identifier (in the EPR) and the queryID of the corresponding mobyData input. This step can be repeated as many times as needed until the service execution is done and the result is ready, using a HTTP POST operation, with the appropriate HTTP headers set, to periodically request for execution status information. Additional notification information (i.e. "step 2: sequence alignment done") can be included to report progress status from the requested service to the client.

### *Request service execution results*

When the client receives a status from the service that indicates that the execution is completed, the client can ask the service for the result by sending a message containing the asynchronous batch-call identifier and queryID to the GetResourceProperty/GetMultipleResourceProperties WSRF operation for the properties in the "result" group.

The response from this operation call will contain WS Resource properties whose formats are MOBY compatible and will include, any MobyExceptions that might have occurred. These properties are all available until the resource is no longer available. A resource becomes unavailable when the service either by its own initiative cleans the result or after a client explicitly asks that the resource is cleaned.

### *The client destroys the session*

Once the client has retrieved the results of all jobs and does not wish to retrieve them again, it should destroy the session created during step 1 above. This is done by POSTing to the service a message containing the EPR to the Destroy WSRF operation.

## Asynchronous Messages

Service execution requests are the only messages that will follow exactly the normal BioMoby standard. All other messages will be WSRF messages.

Regarding error-handling: naturally, it is possible to get other HTTP related errors that we don't specify here.

When using the standard WSRF operations it is possible to get the standard WSRF faults. We give a general example in the [appendix](#) to show readers how such a fault might look like. In this section we list the documented WSRF faults in each situation. For more information about WSRF operations and faults, we again refer the reader to the official WSRF documentation.

## Messages

### 1. Requesting asynchronous service execution:

- This message is identical to the BioMoby XML request to synchronous HTTP POST services.

POST to web service url (either as raw data or using the parameter name 'data'):

```
<?xml version="1.0" encoding="UTF-8"?>
<moby:MOBY xmlns:moby="http://www.biomoby.org/moby">
  <moby:mobyContent>
    <moby:mobyData queryID='queryid00'>
      <!--Standard BioMoby XML for Input -->
    </moby:mobyData>
    <moby:mobyData queryID='queryid01'>
      <!--Standard BioMoby XML for Input -->
    </moby:mobyData>
  </moby:mobyContent>
</moby:MOBY>
```

### 2. Accepted asynchronous requests:

- The service recognizes the request and communicates to the client that its request was accepted. The services then sends a standard WSA compliant message containing two important parts in the EPR:
  - i. Address: This is the address where the EPR is a valid reference to the resulting batch-call. Following the recommendation from WS-Resource 1.2, this address is a composition of the original endpoint plus the resource identified by the job ticket.
  - ii. ReferenceParameters: The ticket representing the service provider identifier for the service execution job. The value of this ticket has no intrinsic meaning. The service provider can choose it to be any legal XML fragment. Clients should not attempt to interpret the value of the ticket; it is simply an identifier and should remain opaque from the point of view of the client.

```

<wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://myserver.com/MyService?asynclD=ID</wsa:Address>
  <wsa:ReferenceParameters>
    <mobyws:ServiceInvocationId xmlns:mobyws="http://biomoby.org/">
      ID
    </mobyws:ServiceInvocationId>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>

```

This EndpointReference is included in the HTTP header, as 'moby-wsrf', in all subsequent messages and is used to refer to the batch-call.

### 3. Polling for service status:

- We assume a polling model where the client queries the service asking for the status of its request. The client makes the polling requests by sending a message to the GetResourceProperty or GetMultipleResourceProperties WSRF operations of the service. The structure of the polling message is an element labelled as a WSA reference parameter with the asynchronous job "ticket" (in the HTTP header) and the original request query identifiers encoded as QNames associated to mobyws namespace (properties mobyws:status\_queryID) in the WSRF message POSTed to the web service (either as raw data or name-value pair, where the name is "data")

#### *GetResourceProperty*

In the:

- HTTP Header:
 

```

<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
  <wsa:Action
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetResourceProperty/GetResourcePropertiesRequest
  </wsa:Action>
  <wsa:To
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    wsu:Id="To">http://myserver.com/MyService</wsa:To>
  <mobyws:ServiceInvocationId
    xmlns:mobyws=http://biomoby.org/
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    wsa:IsReferenceParameter="true"> ID
  </mobyws:ServiceInvocationId></mobyws:moby-wsrf>

```

- HTTP POST:
 

```
<GetResourceProperty
  xmlns="http://docs.oasis-open.org/wsrf/rp-s"
  xmlns:mobyws="http://biomoby.org/">
  mobyws:status_queryId00
</GetResourceProperty>
```

### GetResourceProperties

In the:

- HTTP Header:
 

```
<mobyws:moby-wsrf xmlns:mobyws="http://biomoby.org/">
<wsa:Action
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetMultipleResourceProperties/GetMultipleResourcePropertiesRequest
</wsa:Action>
<wsa:To
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  wsu:Id="To">http://myserver.com/MyService</wsa:To>
<mobyws:ServiceInvocationId
  xmlns:mobyws="http://biomoby.org/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  wsa:IsReferenceParameter="true">
  ID
</mobyws:ServiceInvocationId>
</mobyws:moby-wsrf>
```
- HTTP POST
 

```
<GetMultipleResourceProperties
  xmlns="http://docs.oasis-open.org/wsrf/rp-s"
  xmlns:mobyws="http://biomoby.org/">
<ResourceProperty>
  mobyws:status_queryId00
</ResourceProperty>
<ResourceProperty>
  mobyws:status_queryId01
</ResourceProperty>
</GetMultipleResourceProperties >
```

Valid fault messages are ResourceUnknownFault, ResourceUnavailableFault and InvalidResourcePropertyQNameFault.

4. Response from polling service status:
- The response sent from the service includes the current process status for the requested query identifiers.

#### *GetResourceProperty*

In the:

- HTTP header  

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
  <From
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns="http://www.w3.org/2005/08/addressing"
    wsu:Id="From"
  >http://myserver.com/MyService</From>
  <Action
    xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetResourceProperty/GetResourcePropertyResponse
  </Action>
</mobyws:moby-wsrf>
```
- HTTP Body  

```
<GetResourcePropertyResponse
  xmlns="http://docs.oasis-open.org/wsrf/rp-2">
  <mobyws:status_queryId00
    xmlns:mobyws="http://biomoby.org/"
    <!--LSAE Analysis Event Block -->
  </mobyws:status_queryId00>
</GetResourcePropertyResponse>
```

#### *GetMultipleResourceProperties*

In the:

- HTTP header  

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
  <From
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns="http://www.w3.org/2005/08/addressing"
    wsu:Id="From"
  >http://myserver.com/MyService</From>
```

```

<Action
  xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetMultipleResourceProperties/GetMultipleResourcePropertiesResponse
</Action>
</mobyws:moby-wsrf>

```

- HTTP Body

```

<GetMultipleResourcePropertiesResponse
  xmlns="http://docs.oasis-open.org/wsrf/rp-2">
  <mobyws:status_queryId00>
    <!--LSAE Analysis Event Block -->
  </mobyws:status_queryId00>
  <mobyws:status_queryId01>
    <!--LSAE Analysis Event Block -->
  </mobyws:status_queryId01>
</GetMultipleResourcePropertiesResponse>

```

Valid fault messages are ResourceUnknownFault, ResourceUnavailableFault and InvalidResourcePropertyQNameFault.

## 5. Requesting the result:

- Once the client knows that the service execution is completed and that the result is ready, it should retrieve results by sending a message to the GetResourceProperty or GetMultipleResourceProperties WSRF operations in the server. The message structure is the same as in the step before, but in this case asking for mobyws:result\_queryID properties (encoded as QNames associated to mobyws namespace).

### GetResourceProperty

In the:

- HTTP Header:

```

<mobyws:moby-wsrf xmlns:mobyws="http://biomoby.org/">
<Action
  xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetResourceProperty/GetResourcePropertyRequest
</Action>
<To
  xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns="http://www.w3.org/2005/08/addressing"
  wssu:Id="To">http://myserver.com/MyService</To>
<mobyws:ServiceInvocationId
  xmlns:mobyws="http://biomoby.org/"
  wsa:IsReferenceParameter="true">

```

*ID*

```
</mobyws:ServiceInvocationId>  
</mobyws:moby-wsrf>
```

- HTTP POST:

```
<GetResourceProperty  
  xmlns="http://docs.oasis-open.org/wsrf/rp-s"  
  xmlns:mobyws="http://biomoby.org/">  
  mobyws:result_queryId00  
</GetResourceProperty>
```

### *GetResourceProperties*

In the:

- HTTP Header:

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>  
<Action  
  xmlns="http://www.w3.org/2005/08/addressing">  
http://docs.oasis-open.org/wsrf/rpw-2/GetMultipleResourceProperties/GetMultipleResourcePropertiesRequest  
</Action>  
<To  
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"  
  xmlns="http://www.w3.org/2005/08/addressing"  
  wsu:Id="To">http://myserver.com/MyService</To>  
<mobyws:ServiceInvocationId  
  xmlns:mobyws="http://biomoby.org/"  
  wsa:isReferenceParameter="true">  
  ID  
</mobyws:ServiceInvocationId>  
</mobyws:moby-wsrf>
```

### HTTP POST

```
<GetMultipleResourceProperties  
  xmlns="http://docs.oasis-open.org/wsrf/rp-s"  
  xmlns:mobyws="http://biomoby.org/">  
<ResourceProperty>  
  mobyws:result_queryId00  
</ResourceProperty>  
<ResourceProperty>  
  mobyws:result_queryId01
```

*</ResourceProperty>*  
*</GetMultipleResourceProperties>*

6. Response from requesting the result:

- The content of the requested `mobyws:result_queryID` properties is a standard BioMoby response message containing the result of the service execution.

Note that if a BioMoby related error occurred during the execution, the BioMoby response will contain empty `mobyData` with the corresponding `mobyException`.

### GetResourceProperty

In the:

- HTTP header

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
<From
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns="http://www.w3.org/2005/08/addressing"
  wsu:Id="From"
>http://myserver.com/MyService</From>
<Action
  xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetResourceProperty/GetResourcePropertyResponse
</Action>
</mobyws:moby-wsrf>
```

- HTTP Body

```
<GetResourcePropertyResponse
  xmlns="http://docs.oasis-open.org/wsrf/rp-2"
  xmlns:mobyws="http://biomoby.org/">
<mobyws:result_queryID00>
  <moby:MOBY xmlns:moby=http://www.biomoby.org/moby/>
    <moby:mobyContent>
      <moby:mobyData queryID='queryID00'>
        <!-- Standard BioMOBY XML for output -->
      </moby:mobyData>
    </moby:mobyContent>
  </moby:MOBY>
</mobyws:result_queryID00>
</GetResourcePropertyResponse>
```

## GetMultipleResourceProperties

In the:

- HTTP header

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
  <From
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns="http://www.w3.org/2005/08/addressing"
    wsu:Id="From"
  >http://myserver.com/MyService</From>
  <Action
    xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/GetMultipleResourceProperties/GetMultipleResourcePropertiesResponse
  </Action>
</mobyws:moby-wsrf>
```

- HTTP Body

```
<GetMultipleResourcePropertiesResponse
  xmlns="http://docs.oasis-open.org/wsrf/rp-2">
  <mobyws:result_queryId00>
    <moby:MOBY xmlns:moby=http://www.biomoby.org/moby'>
      <moby:mobyContent>
        <moby:mobyData queryID='queryId00'>
          <!-- Standard BioMOBY XML for output -->
        </moby:mobyData>
      </moby:mobyContent>
    </moby:MOBY>
  </mobyws:result_queryId00>
  <mobyws:result_queryId01>
    <moby:MOBY xmlns:moby=http://www.biomoby.org/moby'>
      <moby:mobyContent>
        <moby:mobyData queryID='queryId01'>
          <!-- Standard BioMOBY XML for output -->
        </moby:mobyData>
      </moby:mobyContent>
    </moby:MOBY>
  </mobyws:result_queryId01>
</GetMultipleResourcePropertiesResponse>
```

It will be a client task (at API level) to compose a standard BioMoby message from the WSRF response (i.e. to extract the <MOBY/> element).

Valid fault messages are ResourceUnknownFault, ResourceUnavailableFault and InvalidResourcePropertyQNameFault.

7. Destroying the resource for asynchronous service execution:

- After the client has retrieved the results of all query identifiers, it should destroy the resource it has created during the asynchronous service execution request by sending a message to the Destroy WSRF operation in the server. The structure of this message is an element labelled as a WSA reference parameter with the asynchronous job ticket (in the HTTP header).

In the:

- HTTP header

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
<To
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns="http://www.w3.org/2005/08/addressing"
  wsu:Id="To"
>http://myserver.com/MyService</To>
<Action
  xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/ImmediateResourceTermination/DestroyRequest
</Action>
<mobyws:ServiceInvocationId
  xmlns:wsa=http://www.w3.org/2005/08/addressing
  xmlns:mobyws="http://biomoby.org/"
  wsa:IsReferenceParameter="true">
  ID
</mobyws:ServiceInvocationId>
</mobyws:moby-wsrf>
```

- HTTP Body

```
<Destroy
  xmlns="http://docs.oasis-open.org/wsrf/rl-2/"/>
```

8. Response from destroying the resource:

In the:

- HTTP header

```
<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
<From
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
```

```

    xmlns="http://www.w3.org/2005/08/addressing"
    wsu:Id="From"
  >http://myserver.com/MyService</From>
  <Action
    xmlns="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/rpw-2/ImmediateResourceTermination/DestroyResponse
  </Action>
</mobyws:moby-wsrf>

```

- HTTP Body

```

<DestroyResponse
  xmlns="http://docs.oasis-open.org/wsrf/rl-2"/>

```

Accepted fault messages are ResourceUnknownFault, ResourceUnavailableFault and ResourceNotDestroyedFault.

## Appendix A – Glossary

**Batch-call (or session):** This is a service invocation and corresponds to one MOBY/mobyContent message

**Job:** A job is a one of the multiple service execution requests that belongs to a service invocation, and corresponds to one mobyData message.

## Appendix B – WSRF Faults

An example of a WSRF fault:

In the:

- HTTP header

```

<mobyws:moby-wsrf xmlns:mobyws=http://biomoby.org/>
  <wsa:From
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    wsa:Id="From"
  >http://myserver.com/MyService</wsa:From>
  <wsa:Action
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
http://docs.oasis-open.org/wsrf/fault
  </wsa:Action>
</mobyws:moby-wsrf>

```

- HTTP Body

```

<FaultMessage xmlns="http://docs.oasis-open.org/wsrf/bf-2">

```

```
< Timestamp>fault_timestamp</Timestamp>  
< Description>fault_description</Description>  
</FaultMessage>
```

These errors come as standard HTTP 200 responses with WSRF specific fault information embedded within the response. All WSRF faults are extended from the standard WSRF BaseFault using the XML Schema extension mechanism<sup>vii</sup>.

---

<sup>i</sup> <http://biomoby.org>

<sup>ii</sup> [http://biomoby.open-bio.org/CVS\\_CONTENT/moby-live/Docs/MOBY-S\\_API/InputMessage.html](http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Docs/MOBY-S_API/InputMessage.html)

<sup>iii</sup> [http://biomoby.open-bio.org/CVS\\_CONTENT/moby-live/Docs/MOBY-S\\_API/OutputMessage.html](http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Docs/MOBY-S_API/OutputMessage.html)

<sup>iv</sup> <http://tinyurl.com/moby-async-proposal>

<sup>v</sup> [http://biomoby.open-bio.org/CVS\\_CONTENT/moby-live/Docs/asyncDocs/BioMOBY Asynchronous Service Specification v2.4.2.pdf](http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Docs/asyncDocs/BioMOBY Asynchronous Service Specification v2.4.2.pdf)

<sup>vi</sup> <http://www.omg.org/cgi-bin/doc?dtd/2005-04-01>

<sup>vii</sup> [http://docs.oasis-open.org/wsr/wsr/ws\\_base\\_faults-1.2-spec-os.pdf](http://docs.oasis-open.org/wsr/wsr/ws_base_faults-1.2-spec-os.pdf)