Exception Reporting in MOBY-S

RFC #1863

INB-05/01

(v 2.02 November 3rd 2005)

Contributions:

Sergio Ramírez
Enrique de Andrés Saiz
Johan Karlsson
José-María Fernández
Antonio J. Pérez
Martin Senger
Coordinador: Oswaldo Trelles, David González-Pisano

National Bioinformatics Institute (INB)

Spain

Madrid, November 3rd 2005

1. - Preliminaries

This document contains the INB[1] proposal for exception reporting in bioMoby. The proposal was discussed at the INB Meeting in Málaga (July, 2005) with the participation of Martin Senger and Edward Kawas, and in the BioMOBY mailing lists during summer 2005.

This proposal aims to contribute to the standardisation of exception reporting in bioMOBY by supplying a more detailed description of common errors and a way to enable uniform notification. Additional information on exceptions will become available, thus allowing the BioMOBY clients to better inform their users about what is going on. Additionally, once a mechanism for asynchronous services becomes standardised by the bioMoby consortium, exceptions related to asynchronous communication can be specified using this proposal.

This document focuses on BioMOBY only (i.e., exceptions rose during service execution that can be reported to the client using a BioMOBY XML payload). In some cases, an exception can be considered critical and the service not able to transmit a bioMOBY XML to the requesting client, but the underlying protocols still able to rise and exception and inform the client. It's been proposed to handle the exceptions at this transport level using SOAP Fault mechanism, but the details of such mechanism (faultCode, faultStrings, etc) are not covered in this document.

[1] Instituto Nacional de Bioinformática (INB), Spain

2- Current (API v0.86) bioMOBY errors specification

BioMOBY protocol specifies that errors in the service should be reported as an empty object, without additional or specific information about the error (see Table 1).

- In the case of Retrieve calls, failure will be silent and an ***empty object*** of the associated output type will be returned.
- There MUST BE as many mobyData response elements as there were mobyData input elements (if a service can not respond to a specific query for whatever reason, this element may ***be empty!***).

Table 1 - Current BioMOBY specifications to report errors

Current BioMOBY specification also includes a **`serviceNotes`** element (see Table 2)

There are two currently defined child elements of the mobyContent tag in a response message - mobyData, and serviceNotes.
[…]
The serviceNotes block is only loosely defined in this version of the API, and is currently meant to contain human-readable free text. serviceNotes are optional.

Table 2 - Current BioMOBY specification for the `serviceNotes` element

The **`serviceNotes`** element is intended to contain general annotations reported by the service in the response message:

```
<?xml version="1.0" encoding="UTF-8"?>
<moby:MOBY xmlns:moby='http://www.biomoby.org/moby'>
<moby:mobyContent moby:authority='illuminae.com'>
 <moby:serviceNotes>Human readable freetext</moby:serviceNotes>
 <moby:mobyData moby:queryID='a1'>
   <moby:Simple moby:articleName=''>
     <moby:Object namespace='AGI_LocusCode' id='At3g19100'/>
   </moby:Simple>
 </moby:mobyData>
 <moby:mobyData moby:queryID='b2'>
   <moby:Simple moby:articleName=''>
     <moby:Object namespace='AGI_LocusCode' id='At3g19100'/>
   </moby:Simple>
 </moby:mobyData>
</moby:mobyContent>
</moby:MOBY>
```

Table 3 - Current moby request (message format) showing the `serviceNotes` tag location

3- Specification Proposal

3.1. Extending the structure of the serviceNotes tag

BioMOBY has to consider reporting exceptions for every layer of information the service request can contain: whole invocation (`mobyContent`), individual queries (`mobyData`), and single objects provided as input to the service (Simples, Collections, or even Simples inside a Collection).

A structured **`mobyException`** child element under `serviceNotes` is the placeholder for exception reporting information[2]. It is worth to note that other additional children elements can be incorporated in a future under the `serviceNotes` tag (to fully enrich the response message with whatever metadata the service would want to communicate to the client).

The **`mobyException`** tag is used to report exception conditions. Using the mandatory attribute **`severity`**, different types of exceptions can be described:

- **`error`**: Corresponds to fatal errors in a service, which causes running of the program to be terminated. Fatal errors are characterised by containing empty objects.

- **`warning`**: Corresponds to an informative diagnostic message that is issued when a service detects an error or potential problem but continues processing and results are provided.

- **`information`**: Corresponds to a free text message not related with any error, containing information that the service wishes to communicate to the user, i.e., non erroneous informative service messages.

One **`mobyException`** tag is associated to each input tag that needs to raise an exception. The link between the erroneous tag and the exception message is made by referring to the `queryID` and `articleName` of the offending input article. Two optional referrer attributes of the **`mobyException`** tag fulfil this linkage function:

- **`refQueryID`** – refers to the `queryID` of the offending input `mobyData`

- **`refElement`** - refers to the `articleName` of the offending input `Simple` or `Collection`

Referrer attributes in **`mobyException`** are optional. An exception can refer to the whole `mobyContent` input if no referrer attributes are present, to a whole query if only **`refQueryID`** is present, or to a single Simple or Collection input if also its **`refElement`** is present. This way it is possible to report exceptions for every input or combination of inputs to the service.

The **`mobyException`** tag contains two elements to describe the error:

- **`exceptionCode`** - exception value (error code)

- **`exceptionMessage`** - human readable description. The message gives more detailed information, complementing the information given by the exception code.

The set of error codes is based on the recommended exception codes system specified in OMG's LSAE standard[3].

---

[2] The downside is that the definition of serviceNotes has to be changed slightly. This should not be a problem but the corresponding documentation should be updated to inform developers.

[3] Life Sciences Analysis Engine (LSAE) final adopted specification - http://www.omg.org/cgi-bin/doc?dtc/2005-04-01

3.2. Examples

Request service XML package(unchanged)

```
<?xml version="1.0" encoding="UTF-8"?>
<MOBY xmlns="http://www.biomoby.org/moby">
       <mobyContent>
              <mobyData queryID='1'>
                     <Simple articleName = "input1">
                            <!-- BioMOBY Object -->
                     </Simple>
              </mobyData>
              <mobyData queryID="2">
                     <Simple articleName = "input2">
                            <!-- BioMOBY Object -->
                     </Simple>
              </mobyData>
              <mobyData queryID="3">
                     <Simple articleName = "input3">
                            <!-- BioMOBY Object -->
                     </Simple>
              </mobyData>
       </mobyContent>
</MOBY>
```

Response service XML package

```
<?xml version="1.0" encoding="UTF-8"?>
<MOBY xmlns="http://www.biomoby.org/moby">
       <mobyContent>
              <serviceNotes>
                     <mobyException refElement="input1"
                                   refQueryID="1"
                                   severity ="error">
                            <exceptionCode>600</exceptionCode>
                            <exceptionMessage>Unable to execute the service</exceptionMessage>
                     </mobyException>
                     <mobyException refElement="input2"
                                   refQueryID="2"
                                   severity ="warning">
                            <exceptionCode>600</exceptionCode>
                            <exceptionMessage>Service execution had non critical
problems</exceptionMessage>
                     </mobyException>
                     <mobyException refElement="input3"
                                   refQueryID="3"
                                   severity ="information">
                            <exceptionCode>600</exceptionCode>
                            <exceptionMessage>No problems in service execution</exceptionMessage>
                     </mobyException>
                     <Notes>Free text Service Notes</Notes>
              </serviceNotes>
              <mobyData queryID="1">
                     <!-- Error: Empty response -->
                     <Simple articleName="output1"/>
              </mobyData>
              <mobyData queryID="2">
                     <!-- Warning: Normal response -->
                     <Simple articleName="output2">
                            <!-- BioMOBY Object -->
                     </Simple>
              </mobyData>
              <mobyData queryID="3">
                     <!-- Information: Normal response -->
                     <Simple articleName="output3">
                            <!-- BioMOBY Object -->
                     </Simple>
              </mobyData>
       </mobyContent>
</MOBY>
```

Table 4 - Example of using the mobyException tag to report error for *Simple* inputs

3.2. Exception Codes

The following is a list describing the exception conditions, such as overflows and errors resulting from incorrect or unmatched data, which are generated during program execution. The error codes are compatible with the LSAE specification.

*New (BioMOBY specific) error types not included in LSAE specification

Exception codes dealing with analysis data		
Code	*Name*	*Description*
200	UNKNOWN_NAME	Setting input data under a non-existing name, or asking for a result using an unknown name
201	INPUTS_INVALID	Input data are invalid; they do not match with their definitions, or with their dependency conditions[4]
202	INPUT_NOT_ACCEPTED	Used when a client tries to send input data to a job created in a previous call but the server does not any more accept input data
221*	INPUT_REQUIRED_PARAMETER	Service require parameter X
222*	INPUT_INCORRECT_PARAMETER	Incorrect parameter X
223*	INPUT_INCORRECT_SIMPLE	Incorrect input in simple article
224*	INPUT_INCORRECT_SIMPLENB	Service requires two or more simple articles
225*	INPUT_INCORRECT_COLLECTION	Incorrect input in collection article
226*	INPUT_EMPTY_OBJECT	Empty input object
227*	INPUT_INCORRECT_NAMESPACE	Incorrect Namespace in the input object

Exception codes dealing with analysis execution		
Code	*Name*	*Description*
300	NOT_RUNNABLE	The same job has already been executed, or the data that had been set previously do not exist or are not accessible anymore. Life Sciences Analysis Engine Adopted Specification
301	NOT_RUNNING	A job has not yet been started. Note that this exception is not raised when the job has been already finished.
302	NOT_TERMINATED	A job is not interruptible for some reason.

Error codes dealing with analysis metadata		
Code	*Name*	*Description*
400	NO_METADATA_AVAILABLE	There are no metadata available

[4] Taken from LSAE, in BioMOBY this means a generic invalid input error. Other specific invalid input errors listed below

Error codes dealing with notification		
Code	*Name*	*Description*
500	PROTOCOLS_UNACCEPTED	Used when a server does not agree on using any of the proposed notification protocols

General error codes		
Code	*Name*	*Description*
600	INTERNAL_PROCESSING_ERROR	A generic catch-all for errors not specifically mentioned elsewhere in this list
601	COMMUNICATION_FAILURE	A generic network failure
602	UNKNOWN_STATE	Used when a network call expects to find an existing state but failed. An example is an unknown handler representing a Job (unknown Job_ID, typical for WebServices platform)
603	NOT_IMPLEMENTED	A requested method is not implemented. Note that the method in question must exist (otherwise it may be caught already by the underlying protocol and reported differently) - but it has no implementation

Service intrinsic exceptions		
Code	*Name*	*Description*
700*	OK	Service execution was correct[5]
701*	SERVICE_INTERNAL_ERROR	Specific errors from the BioMOBY service[6]

[5] This is a placeholder error code to allow the service to provide a formatted information block in the response
[6] I.e. from Blast 'Check the sequence format; it does not seem to be a nucleotide/Amino acid sequence'

4. Specification (API v0.86) changes

In addition to the above (3) section, the following specification changes are proposed:

Change 1

FROM
In the case of Retrieve calls, failure will be silent and an empty object of the associated output type will be returned.

TO
In the case of ~~Retrieve calls~~ an error, failure should ~~be silent~~ raise an exception and an empty ~~object of the associated output type~~ mobyData block with the appropriate queryID will be returned.

Comments to this change: Exception rising is optional, but should be recommended. The wording in the original API was wrong - the object is not empty, the mobyData block is empty.

Change 2

FROM
*There MUST BE as many mobyData response elements as there were mobyData input elements (if a service can not respond to a specific query for whatever reason, this element may **be empty!**)*

TO
*There MUST BE as many mobyData response elements as there were mobyData input elements (if a service can not respond to a specific query for whatever reason, this element may **be empty,** but an exception could be raised to explain why)*

Comments to this change: Exception rising is optional. The service provider could choose to report the failure or not, but still has to return an empty element.

Change 3

FROM
The serviceNotes block is only loosely defined in this version of the API, and is currently meant to contain human-readable free text. serviceNotes are optional.

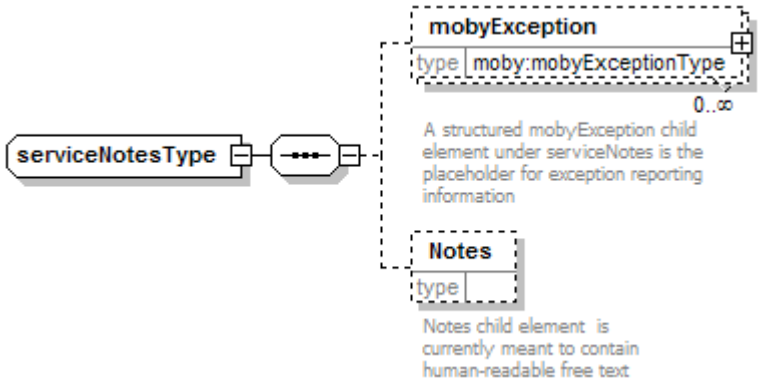
TO
*The serviceNotes block ~~is only loosely defined~~ has been changed to support exception reporting in this version of the API. In addition to the mobyException element for exception reporting, a **Notes** child element ~~and~~ is ~~currently~~ meant to contain human-readable free text. serviceNotes are optional.*

Comments to this change: The original bioMOBY specification states that the serviceNotes element should contain human-readable text. With the addition of an structured XML mobyException element, and to avoid mixed free-text and XML content in serviceNotes, a new Notes child element is added under serviceNotes to preserve its previous functionality.

5. Schema Documentation

complexType **serviceNotesType**

diagram



namespace http://www.biomoby.org/moby

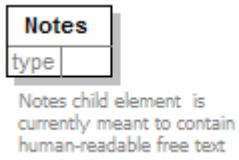
children **mobyException** **Notes**

used by element **mobyContentType/serviceNotes**

source
```
<xs:complexType name="serviceNotesType">
 <xs:sequence>
  <xs:element name="mobyException" type="moby:mobyExceptionType" minOccurs="0" maxOccurs="unbounded">
   <xs:annotation>
    <xs:documentation>A structured mobyException child element under serviceNotes is the placeholder for exception reporting information</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="Notes" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Notes child element  is currently meant to contain human-readable free text</xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
</xs:complexType>
```

element **serviceNotesType/Notes**

diagram



namespace http://www.biomoby.org/moby

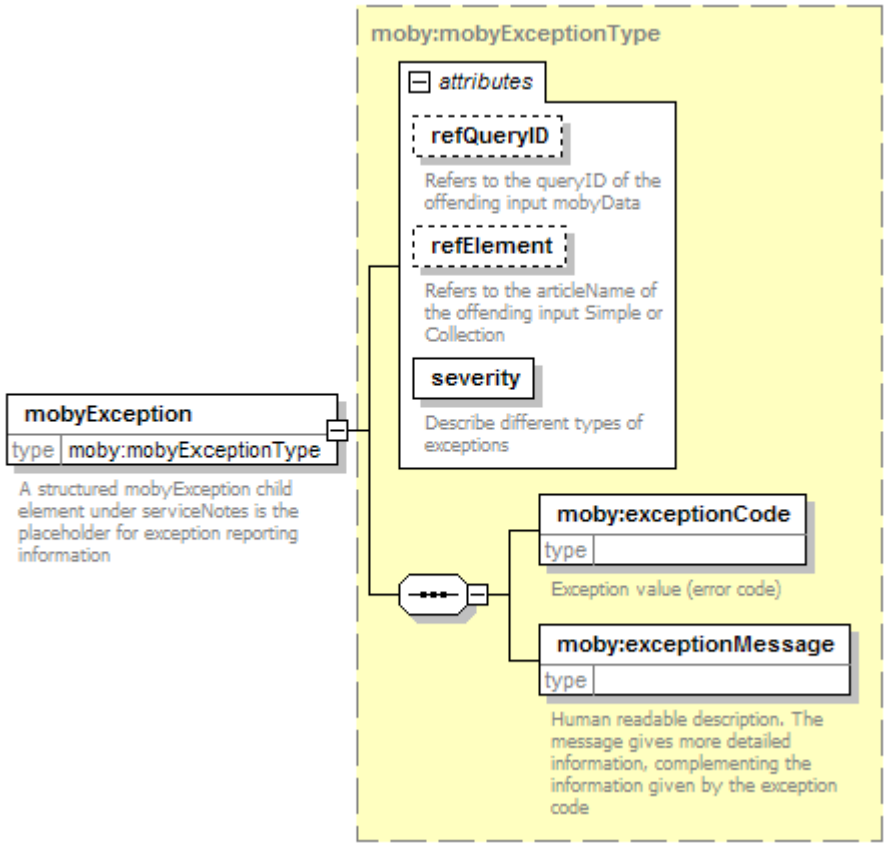
properties isRef 0

annotation documentation Notes child element is currently meant to contain human-readable free text

source
```
<xs:element name="Notes" minOccurs="0">
 <xs:annotation>
  <xs:documentation>Notes child element  is currently meant to contain human-readable free text</xs:documentation>
 </xs:annotation>
</xs:element>
```

element **serviceNotesType/mobyException**

diagram



namespace http://www.biomoby.org/moby

type **moby:mobyExceptionType**

properties
- isRef 0
- content complex

children **moby:exceptionCode** **moby:exceptionMessage**

attributes

Name	Type	Use	Default	Fixed	Annotation	
refQueryID					documentation	Refers to the queryID of the offending input mobyData
refElement					documentation	Refers to the articleName of the offending input Simple or Collection
severity		required			documentation	Describe different types of exceptions

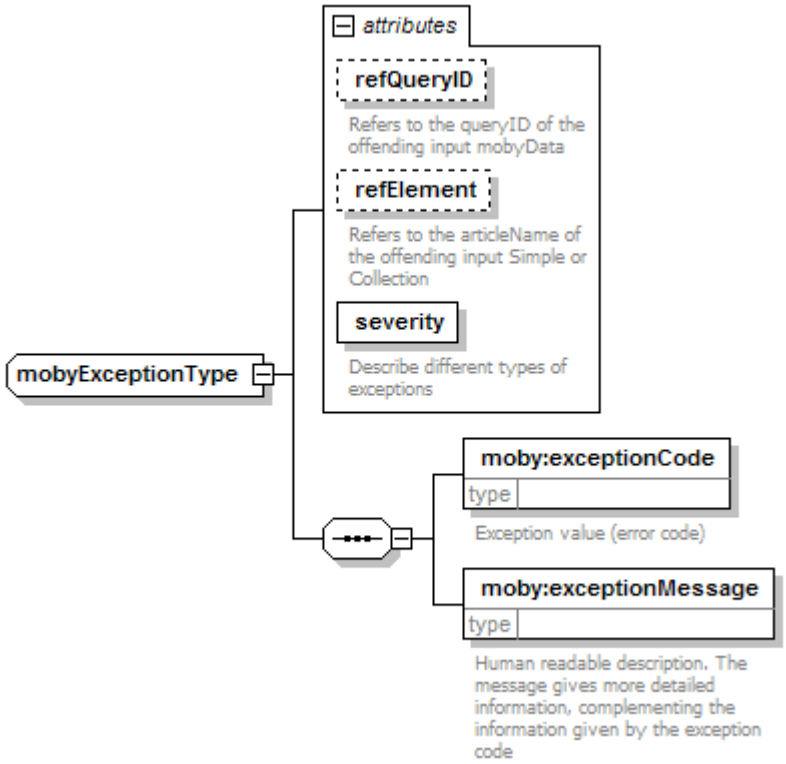
annotation
- documentation A structured mobyException child element under serviceNotes is the placeholder for exception reporting information

source

```
<xs:element name="mobyException" type="moby:mobyExceptionType" minOccurs="0" maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>A structured mobyException child element under serviceNotes is the placeholder for exception reporting information</xs:documentation>
 </xs:annotation>
</xs:element>
```

complexType **mobyExceptionType**

diagram



namespace http://www.biomoby.org/moby

children **moby:exceptionCode** **moby:exceptionMessage**

used by element **serviceNotesType/mobyException**

attributes

Name	Type	Use	Default	Fixed	Annotation	
refQueryID					documentation	Refers to the queryID of the offending input mobyData
refElement					documentation	Refers to the articleName of the offending input Simple or Collection
severity		required			documentation	Describe different types of exceptions

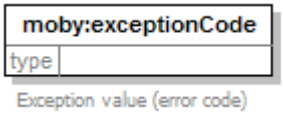
source

```
<xs:complexType name="mobyExceptionType">
 <xs:sequence>
  <xs:element name="exceptionCode">
   <xs:annotation>
    <xs:documentation>Exception value (error code)</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="exceptionMessage">
   <xs:annotation>
    <xs:documentation>Human readable description. The message gives more detailed information, complementing the information given by the exception code</xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="refQueryID">
  <xs:annotation>
   <xs:documentation>Refers to the queryID of the offending input mobyData</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="refElement">
  <xs:annotation>
   <xs:documentation>Refers to the articleName of the offending input Simple or Collection</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="severity" use="required">
  <xs:annotation>
   <xs:documentation>Describe different types of exceptions</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
```

```
      <xs:restriction base="xs:string">
        <xs:enumeration value="error"/>
        <xs:enumeration value="warning"/>
        <xs:enumeration value="information"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

element **mobyExceptionType/exceptionCode**

diagram



namespace http://www.biomoby.org/moby

properties isRef 0

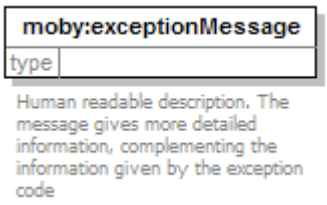
annotation documentation Exception value (error code)

source

```
<xs:element name="exceptionCode">
  <xs:annotation>
    <xs:documentation>Exception value (error code)</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **mobyExceptionType/exceptionMessage**

diagram



namespace http://www.biomoby.org/moby

properties isRef 0

annotation documentation Human readable description. The message gives more detailed information, complementing the information given by the exception code

source

```
<xs:element name="exceptionMessage">
  <xs:annotation>
    <xs:documentation>Human readable description. The message gives more detailed information, complementing the
information given by the exception code</xs:documentation>
  </xs:annotation>
</xs:element>
```