Error handling in BioMOBY

Working Draft GNV5-05/09

(v 1.3)

Contributions:

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

Node GNV-5: Integrated Bioinformatics

University of Malaga

Malaga, 21th July 2005 (3rd Draft)

1.- Preliminaries

This document contains the INB proposals for notification and error handling in bioMoby. This proposal will also enable the implementation of error handling in the INB platform. The proposal was discussed at the INB Meeting in Málaga (July, 2005) with the participation of Martin Senger and Edward Kawas.

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). Naturally, this is a limitation since clients could be interested in the reason for the error, not just simply be informed that an error occurred.

- 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.

BioMOBY version 0.8 includes a new element "*serviceNotes*" (see Table 2), that can contain general annotations reported by the service (at request level).

```
<?xml version="1.0" encoding="UTF-8"?>
<moby:MOBY xmlns:moby='http://www.biomoby.org/moby' xmlns=''>
<moby:mobyContent moby:authority='illuminae.com'>
  <moby:serviceNotes>descriptive content</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 2.- Current moby request (message format) showing the serviceNotes tag location.

First approach: Extending the semantics of the *serviceNotes* tag

The original INB-proposal (also the current –working- implementation at the INB) for error notification and management consists in including a child element under the *serviceNotes* element (associated to each query). It is worth to note that additional children can be incorporated under this scheme (to inform of particular concerns). For example, it is possible to include a mobyData as child to serviceNotes to inform about the different error situations for multiple mobyData in the same queryID. Unix-like error format has been chosen to report information: the service (when necessary) will issue messages which include the "exception code" (value) severity of the error and a human-readable description. The description will give more detailed information, complementing the information given by the error code.

To avoid collision in tag-interpretation performed by clients and / or with extensions of BioMOBY protocol, a specific namespace is being used (i.e. 'inb'), including their explicit identification (xmlns:inb='http://www.inab.org/errorMessages'). This was necessary to not break the MOBY standard because at the develoment time the INB was not coordinated with the BioMOBY consortium.

An *Exception* tag is used to report error conditions. This tag can describe better the different types of exceptions since it is able to cope both with error conditions and warnings (informative messages that do not terminate the execution of the service).

```
<?xml version='1.0' encoding='UTF-8'?>
<moby:MOBY xmlns:moby='http://www.biomoby.org/moby'
xmlns='http://www.biomoby.org/moby'>
<moby:mobyContent moby:authority='illuminae.com'>
   <moby:serviceNotes>
     <inb:Error xmlns:inb='http://www.inab.org/errorMessages'>
       <inb:mobyData queryID='a1' severity='error'>
         <inb:Value>valor</inb:Value>
         <inb:Description>descripcion</inb:Description>
       </inb:mobyData>
   </moby:serviceNotes>

  <moby:mobyData moby:queryID='a1'/>              [Empty mobyData reporting ERROR]

  <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.- Example of the proposed message interchange for service errors. The “inb” tag is used to avoid tag collision with other extensions, and could be replaced (i.e. “moby”) if the proposal is adopted in the bioMOBY protocol.

Although this proposal is non-invasive with regard to the current bioMoby standard, there are some considerations that need to be addressed in order to reach a more complete, robust and useful protocol:

(a) The *serviceNotes* element should remain available for the purpose it was designed by the BioMOBY consortium.
(b) We can only report errors for the entire output. We cannot report errors for specific objects and return partial results from a service.
(c) Since bioMOBY consortium should approve new proposals, in the meanwhile any proposal should avoid collisions with other equivalent (or not) proposals. Special care should be taken with the definition of new tags to be used in the protocol.

2.- Proposal: Define a new tag (MobyException)

This proposal arose during the INB-meeting (Malaga). We reached the conclusion that it is important to develop a way of reporting errors that permits us to report exceptions for every object that the service can return. To achieve that propose, we agree to create a new tag (called MobyException) which will contain information about the error. Similar to the initial proposal, it contains two elements: Value and Description; containing the error value and its description respectively. The tag can be extended with additional children to inform about more specific error situations. This tag can be associated with a Simple tag or with an entire Collection tag. The idea is that we can report errors for every output of the service, and also errors for every element of the collection.

Advantages

- Does not break existing bioMody clients if they ignore the mobyException tag.
- Allows for a more detailed report of an error than the previous INB approach
- Compatible with the error code in LSAE

Problems

- Break the current BioMOBY way to report errors (empty MobyData)

Details of the proposal

The basic idea is to associate a mobyException tag to every Simple/Collection tag that need to report an error. The link between Simple and error message can be achieved by using the same articleName in both tags (look in table 4 for an example).

Request service XML package (unchanged)

```
<?xml version='1.0' encoding='UTF-8'?>
<MOBY xmlns:moby='http://www.biomoby.org/moby-s'>
       <mobyContent>
              <mobyData queryID='1'>
                     <Simple articlename = 'request'>
                            <--- BioMOBY parameters --->
                     </Simple>
              </mobyData>
              <mobyData queryID='2'>
                     <Simple articlename = 'request'>
                            <--- BioMOBY parameters --->
                     </Simple>
              </mobyData>
              <mobyData queryID='3'>
                     <Simple articlename = 'request'>
                            <--- BioMOBY parameters --->
                     </Simple>
              </mobyData>
       </mobyContent>
</MOBY>
```

Response service XML package

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

<moby:MOBY xmlns:moby='http://www.biomoby.org/moby'>

        <moby:mobyContent moby:authority='chirimoyo.ac.uma.es'>
              <mobyData queryID='1'>
                     <Simple articlename='result'>
                            …
                     </Simple>
              </mobyData>
              <mobyData queryID='2'>
                     <Simple articlename='result'>
                            …
                     </Simple>
                     <mobyException articlename='result' severity='Warning'>
                            <Value>X</Value>
                            <Description>…</Description>
                     </mobyException>
              </mobyData>
              <mobyData queryID='3'>
                     <Simple articlename='result'/>
                     <mobyException articlename='result' severity='Error'>
                            <Value>X</Value>
                            <Description>…</Description>
                     </mobyException>
              </mobyData>
       </moby:mobyContent>
</moby:MOBY>
```

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

When we need to report errors for individuals objects in a collection we can associate the error report tag with this elements in a similar way. For this approach to work, we do need a way to identify a single object inside the collection. Therefore, we suggest that is should be mandatory to set a articleName in every Simple tag inside the collection. Adding an articleName would not disturb existing clients since they would simply ignore the attribute. In table 5, we give an example of how a block could look like where errors are reported inside a collection:

Request service XML package

```
<?xml version='1.0' encoding='UTF-8'?>
<MOBY xmlns:moby='http://www.biomoby.org/moby-s'>
       <mobyContent>
              <mobyData queryID='1'>
                     <Simple articlename = 'request'>
                            <--- BioMOBY parameters --->
                     </Simple>
              </mobyData>
       </mobyContent>
</MOBY>
```

Response service XML package

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

<moby:MOBY xmlns:moby='http://www.biomoby.org/moby'>

        <moby:mobyContent moby:authority='chirimoyo.ac.uma.es'>
              <mobyData queryID='1'>
                     <Collection articlename = 'result'>
                            <Simple articlename='s1'>
                                   …
                            </Simple>
                            <Simple articlename='s2'>
                                   …
                            </Simple>
                            <mobyException articlename='s2' severity='Warning'>
                                   <Value>X</Value>
                                   <Description>Description</Description>
                            </mobyException>
                            <Simple articlename='s3'/>
                            <mobyException articlename='s3' severity='Error'>
                                   <Value>X</Value>
                                   <Description>Description</Description>
                            </mobyException>
                     </Collection>
              </mobyData>
       </moby:mobyContent>
</moby:MOBY>
```

Table 5.- Example of using the mobyException tag to report error for *Collections*. In this example, a simple-input/collection-output service is used. In the case of a collection-input/collection-output service in which each simple in the output collection correspond to one simple in the input collection, the articlename in the input collection must be mandatory (since it is used to match input and output simples).

3.- Implementation Protocol

1) Regarding the error codes found in MobyException, we suggest similar error codes as the ones found in the Life Sciences Analysis Engine (LSAE) standard.
2) Client and service should both refer to the same exception catalogue to maintain coherence in the notification system
3) The client will report (task and object monitors) to the user the following information
    a. Exception code
    b. Exception severity
    c. Exception description (general, from the catalogue or namespace)
    d. Specific information contained in the message (description from the service)

4.- Warning condition (example)

A given service retrieves information about protein interactions from different data sources (a couple of SQL databases and one XML data base). Each database is stored in a different server, including the corresponding DBMS.

In some cases, the service is not able to contact with a particular DBMS. Under this situation, rather than stopping the service execution (severity = ERROR) the service goes ahead to the next server and finally report a message with severity=WARNING (assuming that at least one DBMS responded with useful information).

Even with only partial results, the service is still interesting enough for a client to wish it to continue. These types of situations are expected to arise –for example- when working with collections.

5.- Final comments

This proposal aims to contribute to the standardisation of the error protocol by supplying a more detailed description of the error and a way to enable uniform notification. Although the protocol still needs slight modifications, additional information on errors will become available. For example, once a mechanism for asynchronous services becomes standardised by the bioMoby consortium, errors related to asynchronous communication should be specified.

This document is intended to promote discussion. In our opinion this is a simple issue that could improve the BioMOBY standard.

Error codes

Severity qualifiers for exceptions.

Two qualifiers are provided:

error: Correspond to fatal errors in a service, which causes running of the program to be terminated. Fatal errors are characterised by contain empty objects.

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

Error types

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.

[yellow marks] New (specific) error types not included in LSAE specification

Error codes dealing with analysis data

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.

- Incorrect input in simple article.
- Service requires two simple articles.
- Incorrect input in collection article.
- Empty input object.

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 Service require parameter X.

222 Incorrect parameter X.

Error codes dealing with analysis execution

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

400 NO_METADATA_AVAILABLE There are no metadata available.

Error codes dealing with notification

500 PROTOCOLS_UNACCEPTED Used when a server does not agree on using any of the proposed notification protocols.

General error codes

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 Web Services 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.

621 Service not available.

622 Malformed MOBY response.

Service intrinsic errors.

701 Specific errors from the service: i.e. from Blast 'No hits found' or 'Check the sequence format; it does not seem to be a nucleotide/Amino acid sequence'.

702 'Object not found with the given input X'. i.e. the specified namespace is wrong or does not exist (namespace supplied "SwissProt"; expected "Swiss-Prot").

Client-side detected errors.

800 No access permissions.
801. Quotes exceed.
802. Unreachable service.
803. Service timeout .