API Documentation

Error Reports

Grant Agreement	Health-F5-2008-200787
Acronym	OpenTox
Name	An Open Source Predictive Toxicology Framework
Coordinator	Douglas Connect

Contract No.	Health-F5-2008-200787	
Document Type:	API Documentation	
WP/Task:		
Name	API Documentation on Error Reports	
Document ID:	API Documentation	
Date:	06/20/10	
Status:	First Draft, v1.0	
Organisation:	National Technical University of Athens (NTUA)	
Contributors	Pantelis Sopasakis	NTUA

Distribution:	Developers

Purpose of Document:	

Document History:	1 – Document produced by Pantelis Sopasakis (NTUA)

1. Introduction

The purpose of this documentation is to present in a clear way how exceptional events are reported by the services involved in OpenTox and how these are reports proliferate in this distributed system back to the client that performed the request. Exceptional events may occur mainly due to the following reasons:

1. The client is to blame for the request it submitted. The server's response is accompanied by a 4XX status code.
2. The server is to blame for some internal reason such as database connectivity errors.
3. The server received an error from some other service while acting itself as a client. This exception is accompanied by the status code 502.
4. Authentication or authorization errors occurring if the user provides wrong or no credentials or if the user have insufficient privileges to perform an action.

However the status code cannot reveal the roots of the exception or provide by itself a way to cope with it. This is the motivation for adopting a RESTful API for error reporting in OpenTox.

2. Error Reports

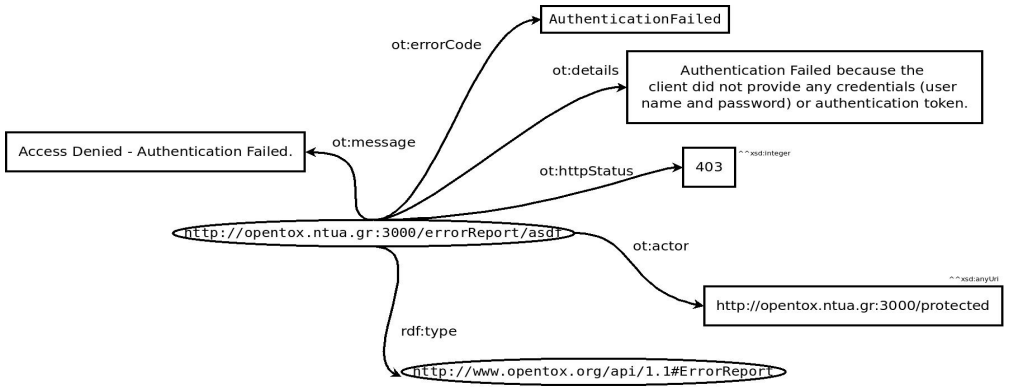
An Error Report consists of the following parameters:

1. **Actor:** The peer that produced the error report, i.e. the URI of the service that throws or catches an exceptional event.
2. **Error Code:** An identifier for the type of the error event. Identifiers for error codes are used to identify error types and not specific instances of such types. Service providers specify a list of standard error codes for each exceptional event and also supply values for the corresponding *message* fields.
3. **Message:** Brief explanatory message for the exceptional event included in the error report. This message is free of technicalities and may also provide a hint to the client about how the problem can be solved.
4. **Details:** Technical details useful for debugging. This message may expose internal information about the service and not be always understandable, but can help developers detect the problem and solve it.
5. **Http Status:** The HTTP status of the response that provides an Error Report.
6. **Trace:** [Optional] If the exceptional event is caused while the service was acting as a client to another service (HTTP status code 502) and the third service returned an error report, this is included in the original error report as the trace of the event.

By convention, only one error report may be returned in a server's response.

2.2 Error Reports in RDF format

Since RDF is the dominant data exchange format adopted in OpenTox, Error Reports are available in all RDF related formats including application/rdf+xml, application/x-turtle, text/x-triple and text/n3-triple. Every error report in RDF has type ot:ErrorReport.



In the figure above you can see an illustration of an Error Report returned in case a client tries to access a protected resource. Below you see the related RDF document:

```
<rdf:Description rdf:about="http://opentox.ntua.gr:3000/errorReport/#1034985908">
<rdf:type rdf:resource="http://www.opentox.org/api/1.1#ErrorReport"/>
<ot:httpStatus>403</ot:httpStatus>
<ot:errorCode>AuthenticationFailed</ot:errorCode>
<ot:actor>http://localhost:3000/bibtex</ot:actor>
<ot:details>Authentication & authorization are needed in order to perform this operation.</ot:details>
<ot:message>Access denied</ot:message>
<dc:identifier>http://opentox.ntua.gr:3000/errorReport/#1034985908</dc:identifier>
</rdf:Description>
```

2.3 Standard Error Codes

2.3.1 Error Codes from NTUA web services

The following tables summarize the various error codes that appear in NTUA services.

Table 1. Authorization and Authentication

Error Code	Explanation
AuthenticationFailed	Authentication failed because the client provided

Error Code	Explanation
	wrong or no credentials (Wrong user name and/or password)
InvalidToken	Access is denied because the client provided a malformed or invalid token. Maybe caused because the token is stale.
UnauthorizedUser	The access is denied because the client provided a valid token (or valid credentials) but do not have sufficient privileges to perform the action.

Table 2. Input – Output Errors

Error Code	Explanation
CommunicationError	A connection to some remote machine is established but data transfer fails. This can be due to packet loss or other transfer layer problems.
ConnectionException	Cannot establish a connection to the remote server. Maybe the remote server is down or other problems related to the Internet connection inhibit the connection.
PublicationError	Internal server error caused because a representation cannot be published to the output stream.
StreamCouldNotClose	Some IO stream cannot be released.
InputStreamUnreadable	The service cannot read from an Input Stream.
FileReadingError	Service could not read from a file because the file either does not exist (while it should) or due to insufficient permissions.
FileWritingError	Could not write to a file. Might be because the file is write-protected or it cannot be created because the directory does not exist.

Table 3. BibTeX service errors

Error Code	Explanation
AuthorNotProvided	Client tried to create a new BibTeX entry without providing the author which is a mandatory parameter.
BibTexNotFoundInDatabase	Client asked for a BibTeX resource that is not found in the database. An HTTP status code 404 (not found) is returned in the response's header.
BibTypeNotSupported	Client attempted to create a new BibTeX entry

Error Code	Explanation
	provide an illegal bib type. This is considered to be a client's error and the corresponding HTTP status code is 400.
ImproperBibTexResource	The client posted to the BibTeX service a resource that could not be parsed.
KnoufBibTexClassNotFound	Internal server error caused because the BibTeX creator which attempted to create an illegal BibTeX entry using a non existing BibTeX class in the Knouf ontology.
KnoufDatatypePropertyNotFound	Service tried to make use of a non existing datatype property of the Knouf ontology.

Table 4. Policy Management Errors

Error Code	Explanation
PolicyCreationError	Error encountered while trying to create a new policy on the policy server by posting a policy XML
PolicyDeletionError	Error encountered while trying to delete a policy.
PolicyUpdateError	Error encountered while trying to update an existing policy on the policy server by applying a PUT to a policy URI.

Table 5. URI Exceptions

Error Code	Explanation
AlgorithmNotFoundInCache	The algorithm URI provided by the client is not found in server's cache.
InvalidModelURI	The URI of the model provided by the client is either malformed or does not exist (See *message* and *details* for more information).
InvalidFeatureURI	The feature URI provided by the client is malformed or does not exist (See *message* and *details* for more information).
InvalidServiceURI	The service tried to post some information to a remote service identified by a malformed URI. This is an internal server error.
InvalidTaskURI	The client was expecting a task URI from the remote service but received some invalid URI or message.