GWD-R.235
NSI-WG

John MacAuley
Tomohiro Kudoh
Chin Guok
August 8, 2017

Error Handling in NSI CS 2.1

Status of This Document

This document provides information to the Grid community on how errors are handled in NSI v 2.1. Grid Forum Working Document (GWD), Recommendation (R).

Copyright Notice

Abstract

This document describes a set of standard error codes for the NSI CS protocol [GFD.212]. Each error code is formaly defined to ensure consistent error reporting through detailed error explanations and example XML elements.

Contents

1. Introduction 2
2. Notational Conventions 2
3. Formal error codes 2
4. Populating the serviceException 4
5. Nested serviceException handling 6
6. Example serviceException elements 8
6.1 00100 – GENERIC_MESSAGE_PAYLOAD_ERROR 8
6.1.1 00101 – MISSING_PARAMETER 8
6.1.2 00102 – UNSUPPORTED_PARAMETER 9
6.1.3 00103 – NOT_IMPLEMENTED 10
6.1.4 00104 – VERSION_NOT_SUPPORTED 10
6.2 00200 – GENERIC_RESERVATION_ERROR 11
6.2.1 00201 – INVALID_TRANSITION 11
6.2.2 00203 – RESERVATION_NONEXISTENT 12
6.3 00300 – GENERIC_SECURITY_ERROR 12
6.3.1 00302 – UNAUTHORIZED 12
6.4 00400 – GENERIC_METADATA_ERROR 14
6.4.1 00405 – DOMAIN_LOOKUP_ERROR 14
6.4.2 00406 – NSA_LOOKUP_ERROR 14
6.4.3 00407 – NO_SERVICEPLANE_PATH_FOUND 15
6.5 00500 – GENERIC_INTERNAL_ERROR 15
6.5.1 00502 – CHILD_SEGMENT_ERROR 15
6.5.2 00503 – MESSAGE_DELIVERY_ERROR 16
6.6 00600 – GENERIC_RESOURCE_UNAVAILABLE 16
6.7 00700 – GENERIC_SERVICE_ERROR 16
6.7.1 00701 – UNKNOWN_STP 16
6.7.2 00703 – LABEL_SWAPPING_NOT_SUPPORTED 17
6.7.3 00704 – STP_UNAVALABLE 17
6.7.4 00705 – CAPACITY_UNAVAILABLE 17
6.7.5 00706 – DIRECTIONALITY_MISMATCH 18
6.7.6 00707 – INVALID_ERO_MEMBER 18
6.7.7 00708 – UNKNOWN_LABEL_TYPE 18
6.7.8 00709 – INVALID_LABEL_FORMAT 19
6.7.9 00710 – NO_TRANSPORTPLANE_PATH_FOUND 19

6.7.10 00800 – GENERIC_RM_ERROR 19
7. Contributors 20
8. Security Considerations 20
9. Glossary 21
10. Intellectual Property Statement 22
11. Disclaimer 22
12. Full Copyright Notice 22
13. References 23

1. Introduction

The NSI Implementation Taskforce, as part of the Global Lambda Integrated Facility (GLIF), was formed to address implementation issues relating to the deployment of NSI CS 2.0 [GFD.212] on the Automated GOLE infrastructure. As part of this effort the taskforce identified the need for consistent NSA error reporting to ease troubleshooting, and to allow applications (including NSA themselves) the ability to use consistent error feedback as part of their error handling and retry strategy.

The NSI CS protocol utilizes the *serviceException* element to convey error information associated with the protocol. This element is commonly used in the “error” message types: SOAP faults, failed messages, and error messages. The structure is relatively flexible and able to handle both simple high-level error information, as well as detailed errors down to the individual attribute value causing a problem. The intention was to provide an expandable structure that could grow with the needs of the protocol and new service definitions as they were defined. However, no formal definition was provided as to how each of the defined protocol errors was mapped into a detailed *serviceException* element. Unfortunately, this left too may decision in the hands of the implementers that resulted in multiple *serviceException* representations for the same basic error.

This living document is an effort to formalize the content of each error and define a consistent error reporting strategy through detailed error explanations and example XML elements for each of standard errors.

2. Notational Conventions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in [RFC 2119].
Words defined in the glossary are capitalized (e.g. Connection). NSI protocol messages and their attributes are written in camel case and italics (e.g. *reserveConfirmed*).

3. Formal error codes

To support the *serviceException* definition the NSI CS protocol defined a hierarchal error code structure to group related error codes together under a common parent error code value. These error codes along with additional error details are populated in the *serviceException* element before returning an error to the requester. There is base set of error codes defined as part of the NSI CS specification that should be considered independent of the services being offered by a network.

Table 1 provides a list of these base protocol error codes.

The *text* element is made up of three parts: “error code: descriptive text: [extra information]”.

- Error code MUST be included.
- Descriptive text MUST be included

- Any other fields are optional.

The information described in column 4 of the table (*variables*) MUST be returned within the XML *variables* element. It is recommended that any other information useful in resolving the issue SHOULD also be returned. This other information will be enclosed in round brackets i.e. ().

Note that the descriptive text element is intended to be human readable information about the error. The *variables* element is intended to be a structured machine readable version.

errorId	***Text***		***variables***
	error code	**Descriptive text**	
00100	GENERIC_MESSAGE_PAYLOAD_ERROR	Illegal message payload.	Include invalid payload elements if available.
00101	MISSING_PARAMETER	Invalid or missing parameter	Include the parameter name that is missing. The value of the invalid payload elements if available should be returned.
00102	UNSUPPORTED_PARAMETER	Provided parameter contains an unsupported value that MUST be processed.	Include the parameter name that is unsupported.
00103	NOT_IMPLEMENTED	Requested feature has not been implemented.	Include the operation or feature that is not implemented.
00104	VERSION_NOT_SUPPORTED	The protocol version requested is not supported.	Return type *protocolVersion* and value of the version requested.
00200	GENERIC_CONNECTION_ERROR	A connection error has occurred.	
00201	INVALID_TRANSITION	Connection state machine is in invalid state for received message.	Include the current state of the state machine.
00203	RESERVATION_NONEXISTENT	Schedule does not exist for *connectionId*.	The *connectionId* in question is returned in the *connectionId* element.
00300	GENERIC_SECURITY_ERROR	A security error has occurred.	
00302	UNAUTHORIZED	Insufficient authorization to perform requested operation.	Include credentials used for authorization evaluation.
00400	GENERIC_METADATA_ERROR	A topology or gerneric path computation error has occurred.	
00405	DOMAIN_LOOKUP_ERROR	Unknown network for requested resource.	Include the resource in question.
00406	NSA_LOOKUP_ERROR	Cannot map networkId to service interface.	Include the networkId in question.
00407	NO_SERVICEPLANE_PATH_FOUND	No service plane path for selected connection segments.	Include source and destination NSA identifiers for the service plane path that could not be found.
00500	GENERIC_INTERNAL_ERROR	An internal error has caused a message processing failure.	
00502	CHILD_SEGMENT_ERROR	Child connection segment error is present.	No local variables are added for this error; however, any variables included in the child segment errors are propagated.
00503	MESSAGE_DELIVERY_ERROR	Failed message delivery to peer NSA.	Include the *providerNSA* (target) of the failed message.
00600	GENERIC_RESOURCE_UNAVAILABLE (non-service specific CA)	A requested resource (s) is not available.	Include the resource in question.
00700	GENERIC_SERVICE_ERROR	A service specific error has occurred.	Reserved for service specific errors as defined by *serviceType* and the corresponding service definition.
00800	GENERIC_RM_ERROR	An internal (N)RM error has caused a message processing failure.	Include information describing the specific (N)RM error.

Table 1 NSI-CS base protocol errors.

As part of an effort to decouple the services offered by a network from the core NSI CS protocol itself, a service-specific parent error code SERVICE_ERROR (00700) was defined for use by individual service specifications. As new services are offered, and existing ones modified, these service-specific errors can be modified as needed with no impact on the core NSI CS protocol. Context for these service-specific errors is provided by the *serviceType* element included in the *serviceException* that is returned when an NSA generates a service-specific error. This *serviceType* element maps into the service definition used for the service request on the failed segment and, in turn, to a detailed description of the service-specific error returned by the *serviceException*.

Table 2 shows the service-specific error codes defined for the point-to-point specific service schema[1]:

errorId	errorDescription	Text	Variables
00701	UNKNOWN_STP	Could not find STP in topology database.	The sourceSTP or destSTP that could not be found.
00703	LABEL_SWAPPING_NOT_SUPPORTED	Label swapping not supported for requested path.	The sourceSTP or destSTP that could not be swapped.
00704	STP_UNAVALABLE	Specified STP already in use.	The sourceSTP or destSTP that is already in use.
00705	CAPACITY_UNAVAILABLE	Insufficient capacity available for reservation.	The capacity value.
00706	DIRECTIONALITY_MISMATCH	Directionality of specified STP does not match request directionality.	The sourceSTP or destSTP with incorrect directionality.
00707	INVALID_ERO_MEMBER	Invalid ERO member detected.	The STP flagged as an invalid member.
00708	UNKNOWN_LABEL_TYPE	Specified STP contains an unknown label type.	The unknown label type.
00709	INVALID_LABEL_FORMAT	Specified STP contains an invalid label.	The invalid STP.
00710	NO_TRANSPORTPLANE_PATH_FOUND	Path computation failed to resolve route for reservation.	

Table 2 – NSI-CS point-to-point service-specific errors.

4. Populating the serviceException

The *serviceException* element in NSI CS 2.1 (see Figure 1) requires an implementation agreement on the encoding representation of any errors and the informational variables returned within the *serviceException*. This can be considered a deficiency in the current protocol definition, specifically with the member *variable* element. The intent was to make the *variable* element self-describing and string based for simplicity. The *namespace* attribute is used to identify the schema of the erred element, the *type* attribute identifies the name of the element, and the *value* element contains the problem parameter. Variables are not in all error messages, and not all variables returned in an error message are parameters in requests. An advantage of this definition is that the variable definition is completely independent of specific parameters in the protocol or service schema definitions.

[1] Service-specific error codes can only be used when the *serviceType* element is populated within the *serviceException* providing context for the error code.

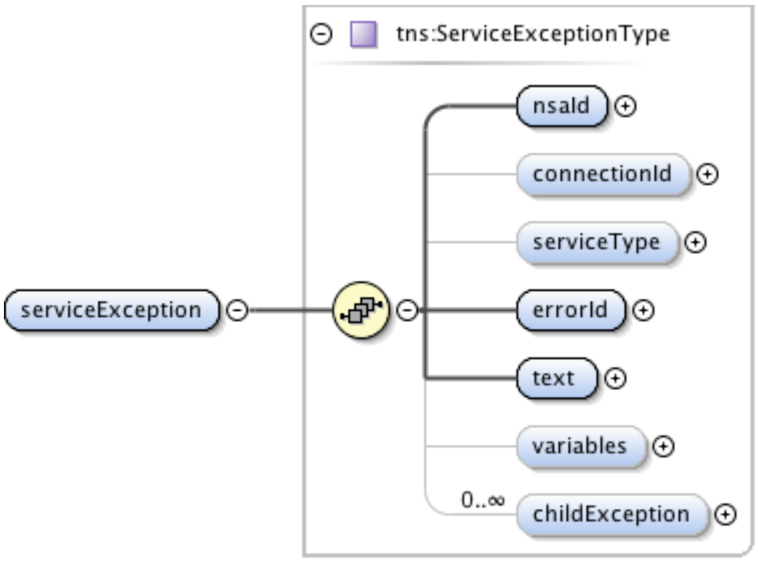

Figure 1: Service Exception structure

Although it is possible to model many errors with this basic structure, it is somewhat hard to manage programmatically. The XML context of the data being model is lost, although the information provided in the *variable* element allows it to be rebuilt. The other issue is that the *value* element is defined as a string, so any data included in value must be serialized as a string. In future versions of the NSI CS protocol a discussion should occur on the topic of changing the definition of the *value* element from string to **xsd:anyType** allowing for more flexibility to include type specific elements within the *value* element. For this document we will discuss populating the *serviceException* element in the context of the existing NSI CS 2.0 specification.

An example is the *serviceException* is reporting that the requested *capacity* of "`5000`" Mb/s is not available on the *sourceSTP* "`urn:ogf:network:example.net:east:stp0`".

In this example, the service-specific 00705 - CAPACITY_UNAVAILABLE error demonstrates the use of the current *serviceException* definition:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00705</errorId>
    <text>CAPACITY_UNAVAILABLE: Insufficient capacity available for reservation
       (5000.0 Mb/s).</text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="capacity">
            <value>5000</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp0</value>
        </variable>
    </variables>
</serviceException>
```

For this example we have the following parameters populated:

- The *nsaId* element contains the NSA generating the error.
- The *connectionId* element contains the reference connection involved in the error.
- The serviceType element identifies the Service Definition used to map the service-specific error identifier.

- The *errorId* element identifies this as a service-specific 00705 - CAPACITY_UNAVAILABLE error.
- The *text* element contains descriptive text for the error. This text should be considered for end-user consumption and not for use programmatically.
- The *variables* element contains two *variable* elements providing additional context to the *serviceException*. The first *variable* models a request parameter called "*capacity*", while the second *variable* models a second request parameter called "*sourceSTP*". The *namespace* attribute identifies the unique schema context of the parameter, while the *type* attribute contains the parameter name itself. The *value* element contains the parameter value associated with *variable* element.

It should be noted that while *sourceSTP* is defined as **xsd:string** in the service-specific schema, *capacity* is defined as an **xsd:long**. The *serviceException* element required a serialization of the original *capacity* parameter from an **xsd:long** to an **xsd:string** for error reporting. Although simple for this example, more complex service elements could require additional specification to serialize. It is important to note that the *namespace* and *type* attributes in variable allow the reconstruction of the original XML context for the parameter.

5. Nested serviceException handling

The *serviceException* element can support nesting of child *serviceException* elements allowing for Aggregator NSA to consolidate error feedback from multiple children NSA. This is an optional feature of the current NSI CS protocol specification, but one that can help a requester agent determine the end-to-end viability of a path more quickly.

In this example we can see *serviceException* from two children NSA (`urn:ogf:network:example.net:2013:nsa:dasher` and `urn:ogf:network:example.net:2013:nsa:dancer`) has been consolidated into a single *serviceException* by the Aggregator NSA (`urn:ogf:network:example.net:2013:nsa:aggregator`). Information relating to the individual child *serviceException* is contained in the *childException* element, while the main *serviceException* body is populated by the Aggregator NSA using information from the *childException* elements.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:aggregator</nsaId>
    <connectionId>urn:uuid:93f679c4-2fa2-42fc-8423-7fe4322c30f9</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00704</errorId>
    <text>STP_UNAVALABLE: Specified STP already in use
(urn:ogf:network:example.net:2013:west:stp1).</text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
            type="destSTP">
            <value>urn:ogf:network:example.net:2013:west:stp1</value>
        </variable>
    </variables>
    <childException>
        <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
        <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
        <serviceType>
            http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE
        </serviceType>
        <errorId>00705</errorId>
        <text>CAPACITY_UNAVAILABLE: Insufficient capacity available for reservation
            (5000.0 Mb/s).</text>
        <variables>
            <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="capacity">
                <value>5000</value>
            </variable>
            <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
                <value>urn:ogf:network:example.net:2013:east:stp0</value>
            </variable>
        </variables>
    </childException>
    <childException>
        <nsaId>urn:ogf:network:example.net:2013:nsa:dancer</nsaId>
        <connectionId>urn:uuid:289e37b6-0b8c-4d66-bd46-cb6eaef456cb</connectionId>
        <serviceType>
            http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE
        </serviceType>
        <errorId>00704</errorId>
        <text>STP_UNAVALABLE: Specified STP already in use
(urn:ogf:network:example.net:2013:west:stp1).</text>
        <variables>
            <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="destSTP">
                <value>urn:ogf:network:example.net:2013:west:stp1</value>
            </variable>
        </variables>
    </childException>
</serviceException>
```

In this case, the Aggregator NSA chose the 00704 – STP_UNAVAILABLE child *childException* as the more critical error for the main body. Which *childException* element is chosen for the body of the parent *serviceException* element is left up to the NSA specific implementation, however, the implementation should consider the parent *serviceException* element may be the only error used by the application.

Inclusion of the *childException* elements is OPTIONAL. The parent *serviceException* MUST be populated, and it is RECOMMENDED that *childException* elements are included when available. It is also RECOMMENDED that the single most critical of the child errors be utilized for the parent *serviceException.*

The *connectionId* within the parent *serviceException* element MUST be the *connectionId* for the reservation in the context of the aggregator, while the *conectionId* for each *childException* element MUST be the *connectionId* in the context of the child NSA.

In some situations it may not make sense for an Aggregator NSA to promote a child *serviceException* due to the nature of the error or unwillingness to choose a specific error from the list of child errors. For example, a RESERVATION_NONEXISTENT error returned from a child NSA with the child's *connectionId* would not make sense to the requesting NSA in the context of the Aggregator if this error were promoted. For these situations a 00502 – CHILD_SEGMENT_ERROR has been defined that allows an Aggregator NSA to populate its *serviceException* element with an error indicating a *childException* element is present holding a more specific error.

In this second example we have a single *serviceException* from the child NSA (`urn:ogf:network:example.net:2013:nsa:dasher`), but in this case the Aggregator NSA (`urn:ogf:network:example.net:2013:nsa:aggregator`) uses the 00502 – CHILD_SEGMENT_ERROR instead of promoting one of the child *serviceException*. Information relating to the individual child *serviceException* is still contained in the *childException* elements.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:aggregator</nsaId>
    <connectionId>urn:uuid:93f679c4-2fa2-42fc-8423-7fe4322c30f9</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00502</errorId>
    <text>CHILD_SEGMENT_ERROR: Child connection segment error is present.</text>
    <childException>
        <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
        <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
        <serviceType>
            http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE
        </serviceType>
        <errorId>00203</errorId>
        <text>RESERVATION_NONEXISTENT: Schedule does not exist for connectionId.
```

```
            (urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b).</text>
        <variables>
            <variable namespace="http://schemas.ogf.org/nsi/2013/12/connection/types"
                type="connectionId">
                <value>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</value>
            </variable>
        </variables>
    </childException>
</serviceException>
```

6. Example serviceException elements

This section provides example *serviceException* elements for each of the defined error codes, providing detailed discussion where required. The NSA implementations MUST follow the mappings of the elements and parameters such that error-reporting consistency can be achieved.

6.1 00100 – GENERIC_MESSAGE_PAYLOAD_ERROR

If the NSI protocol message is malformed in such a way that processing cannot be performed the 00100 – GENERIC_MESSAGE_PAYLOAD_ERROR family of error codes are applicable. *serviceException* elements using this error code will typically be returned in a SOAP fault as a response to the initial operation request as processing through to an NSI operation failure/error is not possible.

The 00100 – GENERIC_MESSAGE_PAYLOAD_ERROR is the parent “catchall” error code used when a more specific error code does not apply. This error is also utilized for situations when the XML for the NSI request is malformed. The *text* element should provide any additional details that are available in either the error text or variable elements.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00100</errorId>
    <text>
        PAYLOAD_ERROR: Illegal message payload (malformed XML).
    </text>
</serviceException>
```

6.1.1 00101 – MISSING_PARAMETER

The 00101 – MISSING_PARAMETER error code is used when a mandatory NSI message parameter is not present in the request. For example, if the *requesterNSA* parameter was missing from the NSI header block in the SOAP message:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00101</errorId>
    <text>
        MISSING_PARAMETER: Invalid or missing parameter (requesterNSA).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/framework/headers"
                type="requesterNSA" />
    </variables>
</serviceException>
```

The *variable* element contains the missing parameter identified by the error. The namespace attribute "`http://schemas.ogf.org/nsi/2013/12/framework/headers`" identifies the header schema as the context for the parameter “`requesterNSA`”. This namespace MUST be included for any NSI header parameters defined in NSI CS 2.0.

The following is an example of a *serviceException* for a missing *connectionId* from a provision request:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00101</errorId>
    <text>
        MISSING_PARAMETER: Invalid or missing parameter (connectionid).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/connection/types"
                type="connectionId" />
    </variables>
</serviceException>
```

As with the previous example, the *variable* element contains the missing parameter identified by the error. The namespace attribute "`http://schemas.ogf.org/nsi/2013/12/connection/types`" identifies the core NSI schema as the context for the parameter "*`connectionId`*". This namespace MUST be included for any core NSI message parameters.

The following is an example of a *serviceException* for a missing service specific schema parameter from a reserve request:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00101</errorId>
    <text>
        MISSING_PARAMETER: Invalid or missing parameter (sourceSTP).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP" />
    </variables>
</serviceException>
```

In this example, the variable element contains the Point-to-Point service specific namespace "`http://schemas.ogf.org/nsi/2013/12/services/point2point`" identifying the context for the "*`sourceSTP`*" parameter. For service specific parameters, the service's namespace MUST be included to identify the parameter context.

6.1.2 00102 – UNSUPPORTED_PARAMETER

The 00102 – UNSUPPORTED_PARAMETER error code is used when a provided parameter contains an unsupported value that MUST be processed. In this example a reserve operation request was received by the NSA with a *directionality* parameter of **Unidirectional**, however, the NSA does not support unidirectional connections, so generates the following *serviceException*:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00102</errorId>
    <text>
        UNSUPPORTED_PARAMETER: Provided parameter contains an unsupported value that MUST
        be processed (directionality).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="directionality">
            <value>Unidirectional</value>
        </variable>
    </variables>
</serviceException>
```

Similar to the previous example, the variable element contains the Point-to-Point service specific namespace "`http://schemas.ogf.org/nsi/2013/12/services/point2point`" identifying the context for the "*`directionality`*" parameter, while the *value* element specifies the unsupported value that caused the error. The namespace attribute MUST be included to identify the parameter context.

6.1.3 00103 – NOT_IMPLEMENTED

The 00103 – NOT_IMPLEMENTED error code can be used to identify a specific operation or feature that has not be implemented. Note there should not be any ambiguity between this error code and the 00102 – UNSUPPORTED_PARAMETER as this is used for parameters.

During the initial deployment period of NSI, some implementations did not support a complete set of operations. In this case we are able to use this error code to identify when an NSA does not support a requested operation. As an example, if a provider NSA did not support the *queryRecursive* operation it would return the following *serviceException*:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00103</errorId>
    <text>
        NOT_IMPLEMENTED: Requested feature has not been implemented.
        (queryRecursive).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/connection/provider"
                type="queryRecursive" />
    </variables>
</serviceException>
```

Within the *variable* element the type attributes identifies the operation as "`queryRecursive`", while the namespace attribute identifies the operation in the context of the provider interface.

A parameter/feature that has not been implemented can be communicated using a similar *serviceException*. The following XML fragment shows a *p2ps* element carried in a *reserve* request. This particular p2ps element has a parameter of type "`protection`" which is a feature not implemented by the provider NSA:

```
<p2ps>
    <capacity>10000</capacity>
    <directionality>Bidirectional</directionality>
    <symmetricPath>true</symmetricPath>
    <sourceSTP>urn:ogf:network:example.net:2013:east:stp0?vlan=3600</sourceSTP>
    <destSTP>urn:ogf:network:example.net:2013:east:stp1?vlan=3600</destSTP>
    <parameter type="protection">PROTECTED</parameter>
</p2ps>
```

The provider NSA to identify the feature that has not been implemented would generate the following *serviceException*:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00103</errorId>
    <text>
        NOT_IMPLEMENTED: the requested feature has not been implemented
        (protection).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="protection" />
        </variable>
    </variables>
</serviceException>
```

This error indicates the "`protection`" feature is not implemented, not that the "`PROTECTED`" is not support otherwise a 00102 – UNSUPPORTED_PARAMETER would have been more appropriate.

6.1.4 00104 – VERSION_NOT_SUPPORTED

The 00104 - VERSION_NOT_SUPPORTED error code is used to identify when an NSI message is received with an unsupported *protocolVersion* element within the NSI header. The version of the

NSI protocol is decoupled from the NSI schema namespace (version of WSDL, XSD, etc.) to allow for behavioral changes in the protocol without needing to revise the wire line protocol.

In this example a requester NSA has sent an NSI CS 2.1 protocol message to a provider NSA that only supports version 2.0 of the protocol.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00104</errorId>
    <text>
        VERSION_NOT_SUPPORTED: The protocol version requested is not supported (version
        2.1).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/framework/headers"
                type="protocolVersion">
            <value>2.1</value>
        </variable>
    </variables>
</serviceException>
```

The *variable* element MUST hold the protocol version requested. The *protocolVersion* element is defined in the NSI header so here the NSI header namespace is used to qualify the "`protocolVersion`" variable. It should be noted that this error is only sent back by the provider agent as a SOAP fault because it cannot be sure that the "failed" or "error" message in the protocol version conforms to that of the requester.

6.2 00200 – GENERIC_RESERVATION_ERROR

The 00200 - GENERIC_RESERVATION_ERROR family of error codes report connection related errors such as an invalid *connectionId* parameter, and invalid states for received operations. The 00200 - GENERIC_RESERVATION_ERROR is the parent "catchall" error code used when a more specific error code does not apply. Provide any additional details that are available in either the error text or variable elements.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <errorId>00200</errorId>
    <text>
        GENERIC_RESERVATION_ERROR: A connection error has occurred.
    </text>
</serviceException>
```

6.2.1 00201 – INVALID_TRANSITION

The 00201 - INVALID_TRANSITION error code is used to identify when a provider NSA has received an NSI operation request on a connection, but that connection is not in the correct state for the received operation.

For example, a provider NSA is creating a connection "`urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b`" that is currently in the "`ReserveHeld`" state. If a requester NSA sends a *provision* operation for this connection, the provider NSA should generate the following *serviceException*:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <errorId>00201</errorId>
    <text>
        INVALID_TRANSITION: Connection state machine is in invalid state for received
        message (ReserveHeld).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/connection/types"
```

```
                type="reservationState">
            <value>ReserveHeld</value>
        </variable>
    </variables>
</serviceException>
```

The *variable* element MUST hold the current state of the connection. The *reservationState* element is defined in NSI connection types schema, so this namespace is used to qualify the "`reservationState`" variable.

6.2.2 00203 – RESERVATION_NONEXISTENT

The 00203 - RESERVATION_NONEXISTENT error code is used to identify when a schedule does not exist for the *connectionId* specified in an NSI operation request. For example, the following serviceException would be generated for a provision operation on connectionId "`urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b`" that does not exist on the provider NSA.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <errorId>00203</errorId>
    <text>
        RESERVATION_NONEXISTENT: Schedule does not exist for connectionId
        (urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b).
    </text>
</serviceException>
```

The requested *connectionId* is reported in the *connectionId* element of the *serviceException* and not using a *variable* element.

6.3 00300 – GENERIC_SECURITY_ERROR

In some situations a general security issue may occur that is not directly related to Authorization of an end-user or peer NSA. In these situations the generic 00300 – GENERIC_SECURITY_ERROR MUST be used to communicate the issue. Any applicable error text SHOULD be provided for troubleshooting.

For non-specific security errors the following *serviceException* is applicable:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00300</errorId>
    <text>
        GENERIC_SECURITY_ERROR: A security error has occurred (Error during certificate path
        validation: timestamp check failed).
    </text>
</serviceException>
```

6.3.1 00302 – UNAUTHORIZED

Authorization in NSI is left for deployment specific implementation so any errors generated, as the result of an authorization failure, will also be deployment specific. Here are a few examples of how the *serviceException* is used to send authorization failure information back to the RA.
If the PA is authorizing the RA by using the subject DN of the X.509 certificate from the TLS session, the following *serviceException* can be sent back when an authorization failure occurs:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00302</errorId>
    <text>
        UNAUTHORIZED: Insufficient authorization to perform requested operation.
    </text>
    <variables>
        <variable type="urn:ogf:nsi:security:attr:realm">
            <value>urn:ogf:network:example.net:2013:nsa:dasher</value>
        </variable>
        <variable type="subject">
            <value>OU=Domain Control Validated, CN=aggregator.example.net</value>
        </variable>
        <variable type="issuer">
            <value>C=US, ST=Arizona, L=Scottsdale, O=Example.net, OU=Cert Repository,
                   CN=Secure Certificate Authority</value>
        </variable>
        <variable type="error">
            <value>The client certificate does not match the defined cert
constraints</value>
        </variable>
    </variables>
</serviceException>
```

In this example, the 00302 – UNAUTHORIZED error code was used to identify the authorization failure. The realm variable MUST be returned to identify the enforced realm that produced the authorization failure. In this case, the local NSA is performing the authorization enforcement. The realm is also used as a context for the additional returned variables if they do not contain a *namespace* attribute. In this example the authorization specific parameters were also returned identifying the DN that failed authorization. More specific error messages can be returned if available.

If the PA is using the <*sessionSecurityAttr*> element for end user authorization, then the matching realm variable MUST be returned to identify the authorization realm that produced the failure, and any authorization specific error information. In the example below, we can see the original OAuth specific <*sessionSecurityAttr*> element passed to the PA within the NSI header for a *reserve* request:

```
<sessionSecurityAttr type="urn:ogf:nsi:security:attr:realm"
         name="http://idp.example.net/oauth">
    <saml:Attribute Name="access_token"
           NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xsd:string">
            2YotnFZFEjr1zCsicMWpAA
        </saml:AttributeValue>
    </saml:Attribute>
</sessionSecurityAttr>
```

The following would then be the *serviceException* generated for an authentication failure in the http://idp.example.net/oauth security realm.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <serviceType>
        http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE
    </serviceType>
    <errorId>00302</errorId>
    <text>
        UNAUTHORIZED: Insufficient authorization to perform requested operation.
    </text>
    <variables>
        <variable type="urn:ogf:nsi:security:attr:realm">
            <value>http://idp.example.net/oauth</value>
        </variable>
        <variable type="error">
            <value>invalid_token</value>
        </variable>
        <variable type="error_description">
            <value>Supplied token is invalid</value>
        </variable>
        <variable type="error_uri">
            <value>http://idp.example.net/oauth/errors/invalid_token.html</value>
        </variable>
    </variables>
</serviceException>
```

6.4 00400 – GENERIC_METADATA_ERROR

The 00400 - GENERIC_METADATA_ERROR error code is a family of codes identifying reservation errors associated with topology and path computation. This is a generic error that is only used when a more specific creation error is not available.

6.4.1 00405 – DOMAIN_LOOKUP_ERROR

The 00405 - DOMAIN_LOOKUP_ERROR error code is used to identify when an error has occurred locating meta data associated with the *networkId* of the specified resource. This typically occurs when the *networkId* of an STP specified in a reservation does not map to a valid NSA description document or topology.

For example, an aggregator NSA attempts to lookup the *networkId* of STP "`urn:ogf:network:example.net:2013:south:stp5?vlan=1790`" but is unable to find the NSA description document or topology document associated with the *networkId* "`urn:ogf:network:example.net:2013:south`":

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00405</errorId>
    <text>
        DOMAIN_LOOKUP_ERROR: Unknown network for requested resource
        (urn:ogf:network:example.net:2013:south:stp5?vlan=1790).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
            type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:south:stp5?vlan=1790</value>
        </variable>
    </variables>
</serviceException>
```

6.4.2 00406 – NSA_LOOKUP_ERROR

The 00406 - NSA_LOOKUP_ERROR error code is used to identify when an error is encountered as the result of invalid or missing discovery meta data. This occurs when the NSA description document of the target NSA does not support the needed version of the CS service interface. See "00405 -" if the *networkId* of the resource matches no meta data documents.

For example, an aggregator NSA attempts to lookup the CS interface associated with the *networkId* of STP "`urn:ogf:network:example.net:2013:south:stp4?vlan=1782`" but while finding the NSA Description document, it is unable to find the required CS service interface:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00406</errorId>
    <text>
        NSA_LOOKUP_ERROR: Cannot map networkId to service interface
        (urn:ogf:network:example.net:2013:south).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
            type="networkId">
            <value>urn:ogf:network:example.net:2013:south</value>
        </variable>
    </variables>
</serviceException>
```

6.4.3 00407 – NO_SERVICEPLANE_PATH_FOUND

00407 - NO_SERVICEPLANE_PATH_FOUND No service plane path for selected connection segments. Include source and destination NSA identifiers for the service plane path that could not be found.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <errorId>00407</errorId>
    <text>NO_SERVICEPLANE_PATH_FOUND: No control plane path for selected connection
segments.</text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/framework/headers"
                type="providerNSA">
            <value>urn:ogf:network:example.net:2013:nsa:dancer</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
            type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:south:stp5?vlan=1790</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
            type="destSTP">
            <value>urn:ogf:network:example.net:2013:south:stp6?vlan=1790</value>
        </variable>
    </variables>
</serviceException>
```

6.5 00500 – GENERIC_INTERNAL_ERROR

The 00500 - GENERIC_INTERNAL_ERROR error code reflects an error internal to the NSA. As each NSA may have distinct error codes due to implementation decisions, the 00500 - GENERIC_INTERNAL_ERROR is the parent “catchall” error code used and can include a NSA specific error code.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <errorId>00500</errorId>
    <text>
        GENERIC_INTERNAL_ERROR: An internal error has caused a message processing failure.
(Connection to internal database has been lost).
    </text>
</serviceException>
```

6.5.1 00502 – CHILD_SEGMENT_ERROR

00502 – CHILD_SEGMENT_ERROR is used by an Aggregator NSA to identify one or more child connection segments has returned an error. This error can be used if the Aggregator does not compute a local error as a result of the operation. No local variables are added for this error; however, any variables included in the child segment errors are propagated.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:aggregator</nsaId>
    <connectionId>urn:uuid:93f679c4-2fa2-42fc-8423-7fe4322c30f9</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00502</errorId>
    <text>CHILD_SEGMENT_ERROR: Child connection segment error is present.</text>
    <childException>
        <nsaId>urn:ogf:network:example.net:2013:nsa:dancer</nsaId>
        <connectionId>urn:uuid:289e37b6-0b8c-4d66-bd46-cb6eaef456cb</connectionId>
        <serviceType>
            http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE
        </serviceType>
        <errorId>00203</errorId>
        <text>RESERVATION_NONEXISTENT: Schedule does not exist for connectionId
              (urn:uuid:289e37b6-0b8c-4d66-bd46-cb6eaef456cb).</text>
        <variables>
            <variable namespace="http://schemas.ogf.org/nsi/2013/12/connection/types"
                type="connectionId">
                <value>urn:uuid:289e37b6-0b8c-4d66-bd46-cb6eaef456cb</value>
            </variable>
        </variables>
    </childException>
</serviceException>
```

6.5.2 00503 – MESSAGE_DELIVERY_ERROR

00503 – MESSAGE_DELIVERY_ERROR is used when an NSA has encountered an MTL or other communication related error (connection timeouts, no route to host, etc.) while attempting to send an NSI message to a peer NSA. Include the providerNSA (target) of the failed message.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:aggregator</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00503</errorId>
    <text>MESSAGE_DELIVERY_ERROR: Failed message delivery to peer NSA.
        (urn:ogf:network:example.net:2013:nsa:dasher).</text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/framework/headers"
            type="providerNSA">
            <value>urn:ogf:network:example.net:2013:nsa:dasher</value>
        </variable>
    </variables>
</serviceException>
```

6.6 00600 – GENERIC_RESOURCE_UNAVAILABLE

The 00600 - GENERIC_RESOURCE_UNAVAILABLE error code is a generic message indicating that one or more of the requested resource is unavailable. This error is generally not used as more service specific errors are defined in the 007xx series codes. The 00600 - GENERIC_RESOURCE_UNAVAILABLE can be considered as the “catchall” error code used for services not (yet) specified or defined in the NSI standard, e.g. compute and storage resources.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <errorId>00600</errorId>
    <text>
        GENERIC_RESOURCE_UNAVAILABLE: Requested resource(s) is unavailable.
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/framework/headers"
                type="Guy"/>
    </variables>
</serviceException>
```

6.7 00700 – GENERIC_SERVICE_ERROR

The 00700 - GENERIC_SERVICE_ERROR family of error codes report service related errors and is used when a more specific error code does not apply.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <errorId>00700</errorId>
    <text>
        GENERIC_SERVICE_ERROR: A service specific error has occurred.
    </text>
</serviceException>
```

6.7.1 00701 – UNKNOWN_STP

Could not find STP in topology database.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00701</errorId>
    <text>UNKNOWN_STP: Could not find STP in topology database
```

```
        (urn:ogf:network:example.net:2013:east:stp9?vlan=1782).</text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp9?vlan=1782</value>
        </variable>
    </variables>
</serviceException>
```

6.7.2 00703 – LABEL_SWAPPING_NOT_SUPPORTED
Label swapping not supported for requested path. Use the renamed LABEL_SWAPPING_NOT_SUPPORTED error

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00703</errorId>
    <text>
        LABEL_SWAPPING_NOT_SUPPORTED: Label swapping not supported for requested
        path.
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp1?vlan=1782</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="destSTP">
            <value>urn:ogf:network:example.net:2013:east:stp7?vlan=1788</value>
        </variable>
    </variables>
</serviceException>
```

6.7.3 00704 – STP_UNAVALABLE
Specified STP already in use.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00704</errorId>
    <text>
        STP_UNAVALABLE: Specified STP already in use
        (urn:ogf:network:example.net:2013:east:stp7?vlan=1788).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp7?vlan=1788</value>
        </variable>
    </variables>
</serviceException>
```

6.7.4 00705 – CAPACITY_UNAVAILABLE
Insufficient capacity available for reservation.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00705</errorId>
    <text>
        CAPACITY_UNAVAILABLE: Insufficient capacity available for reservation
        (5000.0 Mb/s).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
```

```
                type="capacity">
            <value>5000</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp7?vlan=1788</value>
        </variable>
    </variables>
</serviceException>
```

6.7.5 00706 – DIRECTIONALITY_MISMATCH

Directionality of specified STP does not match request directionality. The sourceSTP or destSTP with incorrect directionality.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00705</errorId>
    <text>
        DIRECTIONALITY_MISMATCH: Directionality of specified STP does not match request
        directionality (urn:ogf:network:example.net:2013:east:stp7-in?vlan=1790-1792).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="directionality">
            <value>Bidirectional</value>
        </variable>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="sourceSTP">
            <value>urn:ogf:network:example.net:2013:east:stp7-in?vlan=1790-1792</value>
        </variable>
    </variables>
</serviceException>
```

6.7.6 00707 – INVALID_ERO_MEMBER

If the pathfinder cannot satisfy an ERO then the reservation request fails and a *serviceException* is returned identifying the components of the ERO that caused the failure.

As an example, a requester agent issues a *reserve* request to the ESnet Aggregator NSA identified by *nsaId* `urn:ogf:network:example.net:2013:nsa:aggregator`. The *ero* element contains an intermediate edge *stp* element that is resolvable within NSI topology, but is not associated with an inter-domain SDP. The Aggregator NSA should detect this error during the pathfinding phase and generate a *reserveFailed* response with the following *serviceException* element:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:aggregator</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00708</errorId>
    <text>
        INVALID_ERO_MEMBER: Invalid ERO member detected
        (urn:ogf:network:example.net:2013:north:CLIENT_port_16).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="stp">
            <value>urn:ogf:network:example.net:2013:north:CLIENT_port_16</value>
        </variable>
    </variables>
</serviceException>#
```

6.7.7 00708 – UNKNOWN_LABEL_TYPE

The STP in the request contains a label type that is undefined or unknown. In this example service exception vlan1790 is an incorrectly formatted label: vlan=1790 was expected.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00709</errorId>
    <text>
       UNKNOWN_LABEL_TYPE: Specified STP contains an unknown label type
       (urn:ogf:network:example.net:2013:east:stp5?vlan1790).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="destSTP">
            <value>urn:ogf:network:example.net:2013:east:stp5?vlan1790</value>
        </variable>
    </variables>
</serviceException>
```

6.7.8 00709 – INVALID_LABEL_FORMAT

The STP in the request contains a label that is in a format that is invalid.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00710</errorId>
    <text>
        INVALID_LABEL_FORMAT: Specified STP contains an invalid label
        (urn:ogf:network:example.net:2013:east:stp5?vlan=1795-1790).
    </text>
    <variables>
        <variable namespace="http://schemas.ogf.org/nsi/2013/12/services/point2point"
                type="destSTP">
            <value>urn:ogf:network:example.net:2013:east:stp5?vlan=1795-1790</value>
        </variable>
    </variables>
</serviceException>
```

6.7.9 00710 – NO_TRANSPORTPLANE_PATH_FOUND

Path computation errors are a class of errors associated with failure to find a data path between the specified source and destination STP in the *reserve* request. These errors are assigned the 00710 - NO_TRANSPORTPLANE_PATH_FOUND error code. There are a number of more specific error codes available to identify reservation failures, so the NO_TRANSPORTPLANE_PATH_FOUND error should only be used when a more specific error cannot be found.

The following is an example of a path computation error for a reservation request.

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00710</errorId>
    <text>
       NO_TRANSPORTPLANE_PATH_FOUND: Path computation failed to resolve route for
       reservation.
    </text>
</serviceException>
```

6.7.10 00800 – GENERIC_RM_ERROR

Internal NRM errors are a class of errors associated with failures in the network management software component of the PA, and are assigned the 00800 - GENERIC_RM_ERROR error code. Information associated with this error will typically be NRM specific, and not correctable by the client. Any additional information provided in the *serviceException* will be helpful in reporting the issue for troubleshooting, but will not be usable by the RA for corrective action programmatically.

For internal NRM errors caused by an inter-process communication error while processing an NSI message, the following *serviceException* is applicable:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00800</errorId>
    <text>
       GENERIC_RM_ERROR: An internal NRM error has caused a message processing failure
       (internal communication error).
    </text>
</serviceException>
```

For internal NRM errors caused by invalid internal states or software bugs, the same *serviceException* is applicable but with a change in descriptive text:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00800</errorId>
    <text>
        GENERIC_RM_ERROR: An internal NRM error has caused a message processing failure
        (path computation).
    </text>
</serviceException>
```

Operation timeouts within NSI can be triggered by the MTL in the case of a message delivery timeout, the Coordinator in the case of a missing confirmed/failed/error reply, the PA in the case of a commitTimeout, or the NRM in the case of internal timers triggering operation related issues. The GENERIC_RM_ERROR error code is also used for these NRM related timeout errors.

For internal NRM timeout errors the same *serviceException* is applicable but with a change in descriptive text:

```
<serviceException>
    <nsaId>urn:ogf:network:example.net:2013:nsa:dasher</nsaId>
    <connectionId>urn:uuid:92d54ff8-dec2-4be8-ae9e-3c0244f2c82b</connectionId>
    <serviceType>http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-GOLE</serviceType>
    <errorId>00800</errorId>
    <text>
        GENERIC_RM_ERROR: An internal NRM error has caused a message processing failure
        (internal timeout).
    </text>
</serviceException>
```

7. Contributors

Chin Guok, ESnet
Gerben van malenstein, SURFnet
John MacAuley, ESnet
Tomohiro Kudoh, AIST
Guy Roberts, GÉANT

8. Security Considerations

Security considerations are dealt with in Open Grid forum GWD-R draft-trompert-gwdi-nsi-aa-v04, NSI Authentication and Authorization [NSI AA].

No additional security issues are raised in this document.

9. Glossary

Aggregator (AG)	The Aggregator is an NSA that has more than one child NSA, and has the responsibility of aggregating the responses from each child NSA.
Connection	A Connection is an NSI construct that identifies the physical instance of a circuit in the data plane. A Connection has a set of properties (for instance, Connection identifier, ingress and egress STPs, capacity, or start time). Connections can be either unidirectional or bidirectional.
Connection Service (CS)	The NSI Connection Service is a service that allows an RA to request and manage a Connection from a PA.
Connection Service Protocol	The Connection Service Protocol is the protocol that describes the messages and associated attributes that are exchanged between RA and PA.
Control and Management Planes	The Control Plane and/or Management Plane are not defined in this document, but follow common usage.
Coordinator	The Coordinator function has the role of providing intelligent message and process coordination, this includes tracking and aggregating messages, replies and notifications and the servicing of query requests.
Data Plane	The Data Plane refers to the infrastructure that carries the physical instance of the Connection, e.g. the Ethernet switches that deliver the circuit.
Discovery Service	The NSI discovery service is a web service that allows an RA to discover information about the services available in a PA and the versions of these services.
Inter-Network Topology	This is a topological description of a set of Networks and their transfer functions, and the connectivity between Networks.
Network	A Network is an Inter-Network topology object that describes a set of STPs with a Transfer Function between STPs.
Network Resource Manager (NRM)	The Network Resource Manager owns a set of transport resources and has ultimate responsibility for authorizing and managing the use of these resources. Each NRM is always associated with a single NSA.
Network Services	Network Services are the full set of services offered by an NSA. Each NSA will support one or more Network Services.
Network Service Agent (NSA)	The Network Service Agent is a concrete piece of software that sends and receives NSI Messages. The NSA includes a set of capabilities that allow Network Services to be delivered.
Network Service Interface (NSI)	The NSI is the interface between RAs and PAs. The NSI defines a set of interactions or transactions between these NSAs to realize a Network Service.
Network Services Framework (NSF)	The Network Services framework describes an NSI message-based platform capable of supporting a suite of Network Services such as the Connection Service and the Topology Service.
NSI Message	An NSI Message is a structured unit of data sent between an RA and a PA.
NSI Topology	The NSI Topology defines a standard ontology and a schema to describe network resources that are managed to create the NSI service. The NSI Topology as used by the NSI CS (and in future other NSI services) is described in: GWD-R-P: Network Service Interface Topology Representation [3].
ero	An Explicit Routing Object (ero) is a parameter in a Connection request. It is an ordered list of STP constraints to be used by the inter-Network pathfinder.
Requester/Provider Agent (RA/PA)	An NSA acts in one of two possible roles relative to a particular instance of an NSI. When an NSA requests a service, it is called a Requester Agent (RA). When an NSA realizes a service, it is called a Provider Agent (PA). A particular NSA may act in different roles at different interfaces.
Service Demarcation Point (SDP)	Service Demarcation Points (SDPs) are NSI topology objects that identify a grouping of two Edge Points at the boundary between two Networks.
Service Termination Point (STP)	Service Termination Points (STPs) are NSI topology objects that identify the Edge Points of a Network in the intra-network topology.

Service Plane	The Service Plane is a plane in which services are requested and managed; these services include the Network Service. The Service Plane contains a set of Network Service Agents communicating using Network Service Interfaces.
Service Definition	An XML document that describes the parameters that can specified when requesting a new service.
Simple Object Access Protocol (SOAP)	SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.
Ultimate PA (uPA)	The ultimate PA is a Provider Agent that has an associated NRM.
Ultimate RA (uRA)	The Ultimate RA is a Requester Agent is the originator of a service request.
XML Schema Definition (XSD)	XSD is a schema language for XML.
eXtensible Markup Language (XML)	XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

10. Intellectual Property Statement

11. Disclaimer

12. Full Copyright Notice

13. References

[GFD.212] OGF GFD-I.212, Network Service Interface Connection Service, v2.0.
[NSI AA] OGF GFD draft-gwdi-trompert-nsi_aa-public-comment-v7