**NSI Connection Service Protocol v2.0**

Status of This Document

This document provides information to the Grid community on the NSI Connection Service that operates on the interface between a requesting software agent and the provider software agent. It describes the protocol,statemachine, architecture and associated processes and environment in which software agents interact to deliver the Connection Service. Distribution is unlimited.

Copyright Notice

Copyright © Open Grid Forum (2008-2012). All Rights Reserved.

Trademark

OGSA is a registered trademark and service mark of the Open Grid Forum.

Abstract

This document describes the Connection Service Protocol for the Network Service Interface (NSI). The Connection Service is used to manage connection oriented circuits that transit network providers. The Network Service Interface (NSI) is defined to be the set of protocols and parameters that are used between a software agent requesting a network service and the software agent providing that Network Service. The Connection Service is intended to operate within the Network Service Framework (NSF, GFD.173).

Contents

[1. Overview 3](#_Toc327290167)

[2. Connection Service architecture 4](#_Toc327290168)

[2.1 Layering in NSI 4](#_Toc327290169)

[2.2 Building blocks of the NSI Architecture 4](#_Toc327290170)

[2.3 NSI protocol layer 4](#_Toc327290171)

[2.4 Message Delivery Layer (MDL) 5](#_Toc327290172)

[2.5 Message Transport Layer (MTL) 5](#_Toc327290173)

[3. Connection Service lifecycle 5](#_Toc327290174)

[3.1 Connection Service primitives 5](#_Toc327290175)

[4. Connection Service state machine v2.0 7](#_Toc327290176)

[4.1.1 Ultimate RA/Aggregator State Machine 7](#_Toc327290177)

[4.1.2 Ultimate PA State Machine 9](#_Toc327290178)

[4.2 Requesting a Reservation 10](#_Toc327290179)

[4.3 Provisioning a Connection 10](#_Toc327290180)

[4.4 Releasing the Provisioning state 11](#_Toc327290181)

[4.5 Terminating a Connection 11](#_Toc327290182)

[4.6 Forced end 11](#_Toc327290183)

[4.7 Querying a Connection 11](#_Toc327290184)

[5. NSI Framework Header 11](#_Toc327290185)

[6. NSI Connection Services Primitives 14](#_Toc327290186)

[6.1 Reserve 14](#_Toc327290187)

[6.2 ReserveConfirmed 14](#_Toc327290188)

[6.3 ReserveFailed 14](#_Toc327290189)

[6.4 Provision 15](#_Toc327290190)

[6.5 ProvisionConfirmed 15](#_Toc327290191)

[6.6 ProvisionFailed 15](#_Toc327290192)

[6.7 Release 15](#_Toc327290193)

[6.8 ReleaseConfirmed 16](#_Toc327290194)

[6.9 ReleaseFailed 16](#_Toc327290195)

[6.10 Terminate 16](#_Toc327290196)

[6.11 TerminateConfirmed 16](#_Toc327290197)

[6.12 TerminateFailed 17](#_Toc327290198)

[6.13 Query 17](#_Toc327290199)

[6.14 QueryConfirmed 17](#_Toc327290200)

[6.15 QueryFailed 18](#_Toc327290201)

[6.16 ForcedEnd 18](#_Toc327290202)

[7. Connections: Transport and Service planes 18](#_Toc327290203)

[8. Connection service use of the NSI Topology 19](#_Toc327290204)

[8.1 Representation of NSI Topology 19](#_Toc327290205)

[8.1.1 Introduction to concept of STPs 20](#_Toc327290206)

[8.1.2 Requesting a path using STPs 20](#_Toc327290207)

[8.1.3 Under-specified STPs 20](#_Toc327290208)

[8.1.4 Explicit routing using STPs 20](#_Toc327290209)

[8.2 STPs and Paths 20](#_Toc327290210)

[8.2.1 STP syntax 20](#_Toc327290211)

[8.2.2 Under-specified STPs 21](#_Toc327290212)

[8.2.3 Path 21](#_Toc327290213)

[8.2.4 Path example in an NSA chain 21](#_Toc327290214)

[8.2.5 Path example in an NSA tree 22](#_Toc327290215)

[8.3 Inter-domain vs intra-domain pathfinding 23](#_Toc327290216)

[8.4 Distribution of NSI Topology 24](#_Toc327290217)

[8.4.1 Transport and Service plane relationship 24](#_Toc327290218)

[8.4.2 Bootstrapping Topology Exchange 24](#_Toc327290219)

[8.4.3 Expanding the Topology Exchange 24](#_Toc327290220)

[8.4.4 Update Mechanism 24](#_Toc327290221)

[8.4.5 Policy on Topology Distribution 24](#_Toc327290222)

[9. Security 24](#_Toc327290223)

[9.1 Authentication and Authorization 24](#_Toc327290224)

[9.1.1 Security Requirements 24](#_Toc327290225)

[9.1.2 Message Security 25](#_Toc327290226)

[9.1.3 Authorization 25](#_Toc327290227)

[9.2 Failures and exceptions 25](#_Toc327290228)

[9.2.1 Service plane failures 25](#_Toc327290229)

[9.2.2 Transport plane failures 25](#_Toc327290230)

[10. Appendix A: Best Practices for NSA implementation 26](#_Toc327290231)

[10.1 MDL 26](#_Toc327290232)

[10.2 Message transport error handling 27](#_Toc327290233)

[10.3 ACK handling 28](#_Toc327290234)

[10.4 Guidelines on timeouts: 28](#_Toc327290235)

[10.5 Parallel processing of messages: 28](#_Toc327290236)

[10.6 NTP servers 28](#_Toc327290237)

[11. Appendix B: Service Definitions for Connection Services 28](#_Toc327290238)

[12. Appendix C: NSI topology 29](#_Toc327290239)

[12.1 STP, SDP, Networks, NSAs concepts and syntax 29](#_Toc327290240)

[12.1.1 Possible Improvements 30](#_Toc327290241)

[12.2 Relationship between NSI topology and NML 30](#_Toc327290242)

[12.3 NSI topology best practices 31](#_Toc327290243)

[13. Appendix D: Information Services 31](#_Toc327290244)

[14. Appendix E: Service Definitions for Connection Services 31](#_Toc327290245)

[15. Appendix F: Changes to NSI state machine between v1.1 and v2.0 33](#_Toc327290246)

[15.1 Activated state 33](#_Toc327290247)

[15.2 Unsolicited Messages from PA to RA (Notify) 33](#_Toc327290248)

[15.3 Message Delivery Layer (MDL) 33](#_Toc327290249)

[15.4 Separation of uRA/Aggregator, and uPA State Machines (SM) 33](#_Toc327290250)

[16. Appendix G: SOAP Messaging Examples 34](#_Toc327290251)

[17. Contributors 37](#_Toc327290252)

[18. Glossary 37](#_Toc327290253)

[19. Intellectual Property Statement 39](#_Toc327290254)

[20. Disclaimer 39](#_Toc327290255)

[21. Full Copyright Notice 40](#_Toc327290256)

[22. References 40](#_Toc327290257)

# Overview

1.1 Summary

The NSI protocol is defined by a suite of documents. This recommendation describes the NSI Connection Service Protocol and should be read in conjunction with the NSI Network Services Framework (NSF, GFD.173).

The Network Service Framework (NSF) defines several key architectural elements: a Network, a Network Service, a Network Service Agent (NSA), a Network Service Interface (NSI), and a NSI Protocol. These elements exist in a notional Network Service Plane. The framework describes an environment within which network resources are treated as explicitly manageable objects. Within the framework, these network resources can be selected, allocated, interrogated, and manipulated by software agents on behalf of requesting users.

Network resources and capabilities are presented to the consumer through a set of ‘Network Services’. The Network Services Framework presents a unified model for interacting with these services. Network Services include the ability to create connections, to share topology information, and to do other services needed by a set of federated NSAs.

This document defines a Connection Service protocol to support the reservation, creation, management and removal of Connections.

Where capitalized words are used in this document, these have a formal definition; see the glossary for details.

1.1 Context

Multi-protocol environments

Traditional models of circuit services and control planes adopt a single very tightly defined Data Plane technology, and then hard code the service attributes (signaling parameters) into the control plane protocols. Emerging multiprotocol services will need to leverage many Data Plane technologies and will need to recognize a wider array of service attributes. The NSI supports an abstracted notion of a Connection, and the NSI Service Definition provides a mechanism for specifying service specific constraints or limits on that connection as realized in different Networks. It is up to the NSA path finders and/or the NRM pathfinders to select a path that meets the Service Request and is consistent with the Service Definitions along that path. NSI allows a single Service Plane protocol suite to present different services and different transport capabilities to the user.

Multi-Provider environments

Traditional models of circuit services and control planes assume operation within a single operator environment. NSI is designed to support the creation of Connections that transit several operators. This creates specific requirements for authorization and authentication which are addressed by NSI.

# Connection Service architecture

## Layering in NSI

The NSI protocol is made up of 3 conceptual layers: NSI protocol, NSI Message Delivery Layer and the NSI Transport Layer. The NSI protocol layer is described by the NSI state machines and the NSI messages. The Message Delivery Layer confirms the delivery of messages to immediate adjacent NSAs (known as ‘children’ in the NSI message hierarchy) and aggregates response where. The message transport layer ensures peer-to-peer message delivery.

Figure 2. Conceptual layering in NSA

## Building blocks of the NSI Architecture

The Network Service Agent (NSA) is a software agent that manages associated Network resources. The ultimate Requester Agent uRA is the agent that originates of the Connection request. The ultimate Provider Agent uPA is the agent that delivers the Connection. The ultimate PA is associated with a Network Resource Manager (NRM) which is responsible for delivering the Connection. Where a connection is split out to multiple child agents, the AGgregation agent (AG) is used.

Messages between the uPA and the NRM, while mentioned in this document, do not form a normative part of the NSI CS protocol.

## NSI protocol layer

The Connection Service communicates by exchanging messages between Requester Agent (RA) and Provider Agent (PA) state-machine pairs. These NSI messages and the associated state-machine make up the NSI CS protocol definition. **Error! Reference source not found.** shows an example of the exchange of NSI Connection Service messages over a hierarchy of RA/PA pairs to form a hierarchical tree.

Figure 3. NSI hierarchical tree (remove MDL boxes here?)

When requesting a Connection, a single ultimate RA initiates a hierarchical message exchange by forwarding a Connection request to one or more child NSAs. These NSAs can either handle the request locally (as an ultimate provider) or act as an aggregating NSA and forward the request on to one or more child NSAs.

## Message Delivery Layer (MDL)

Conceptually, MDL resides between the NSI protocol layer and the message transport layer and is responsible for:

1. The “determined” delivery of messages by using mechanisms such as timeout and re-tries.
2. The aggregation of replies from child nodes to indicate if; i. all the children successfully received the message, or ii. if one or children failed to receive the message.

If the MDL is unable to confirm message receipt from all children, it returns a failure. This failure is considered fatal, and the NSA must initiate a state clean up.

For consistency, the MDL conceptual separates the Aggregator and ultimate Provider functions within an NSA, as seen in the lower right NSA in Figure 3.

## Message Transport Layer (MTL)

The transport layer is out of scope of the NSI standard. The purpose of the MTL is to ensure that secure transport between peering NSAs. The recommended transport layer is TCP/IP-HTTP-XML (such as SOAP).

# Connection Service lifecycle

## Connection Service primitives

The Connection Service (CS) protocol is a message based command-response protocol that operates between an RA and a PA.

The NSI CS protocol defines a set of six primitives that provide the control necessary to manage Connections; these are described in Table 1.

|  |  |
| --- | --- |
| **Primitive** | **Description** |
| ***reserve*** *(Request/Confirm/Failed)* | The RA requests the PA to Reserve network resources for a Connection between two STP’s constrained by certain service parameters. |
| ***provision*** *(Request/Confirm/Failed)* | The RA requests the PA to Provision a Reservation (associated with a previous reservation message). |
| ***release*** *(Request/Confirm/Failed)* | The RA request for the PA to de-provision resources without removing the Reservation. |
| ***Activate***  *(Request/Confirm/Failed)* |  |
| ***terminate*** *(Request/Confirm/Failed)* | The RA request for the PA to release the Provisioned resources and terminate the Reservation. |
| ***forcedEnd***  *(Request only)* | This is reported by the PA to the RA to notify that the PA has forced a termination of the Reservation. |
| ***query***  *(Request only)* | Mechanism for either RA or PA to query the other NSA for a set of connection service instances between the RA-PA pair. This message can be used as a status polling mechanism. |

Table : List of primitives

The 3 possible extensions for the message primitives are:

* ***Request*** – The RA sends the request to the PA, for example *reserveRequest*.
* ***Confirm*** - A PA sends this positive operation response message (such as *reserveConfirm*) to the Requester NSA that issued the original request message (*reserveRequest*).
* ***Failed*** - A Provider NSA sends this negative operation response message (such as *reserveFailed*) to the Requester NSA that issued the original request message (*reserveRequest*).

The following figure shows to examples of how message primitives are used to first Reserve and then Provision a Connection. Two modes of Provisioning are supported: Manual and Automatic.

Z:\OGF_NSI\NSI _protocol\CS lifecycle.emf

Figure : Connection Service lifecycle

For brevity of this diagram and the state-machine, the NSI messages are abbreviated as follows:

rsv reserve

prov provision

rel release

term terminate

fcd\_end forcedEnd

rq request

cf confirm

fl failed

# Connection Service state machine v2.0

### Ultimate RA/Aggregator State Machine

The uRA/Aggregator v2.0 state machine is represented below (see Figure 5). Several design decisions reflected in this state machine include:

* *Absence of “Activating” state*. The uRA and Aggregator do not manage any network resources and therefore are unaware when the activating process is initiated. Only when it receives an “activate\_ok.nt” notify message does it know that the activation process has completed.
* *In the “Activated” state, when a provision request is received (“prov.rq”), it will return both a provision confirm (“prov.cf”) as well as an activate notification (“activate\_ok.nt”).* Due to the decoupling of the control and data plane provision processes, both messages (“prov.fc, “activate\_ok.nt”) must be returned to indicate that the reservation is active and the circuit has been setup.
* *In any state, if a failure (“\*.fl”) or forced end (“fcd\_end”) is received, the state machine will transition to the “Terminated” state bypassing the “Terminating” state.* The “Terminating” state is to reflect normal clean up operations when a termination request (“term.rq”) is received, and not part of failure scenarios or when a reservation has expired (i.e. reached it’s end time).

Figure 5. Ultimate RA and Aggregator NSI CS “Oxford” State Machine

These Connection Service message primitives are used to initiate, manage and remove a Connection (the connection life cycle). A single state-machine for both RA and PA describes the state changes and their relationship to messages. The primitives, timing events and NRM responses form inputs and outputs to the state-machine. The connection state can be any of the following:

|  |  |
| --- | --- |
| **State** | **Description** |
| Initial (pseudo state) | the state-machine is not created until a *reserveRequest* is received |
| Reserving | A *reserveRequest* has either been sent or received and the processing is in progress. |
| Reserved | The *reserveRequest* has succeeded and a connection schedule has been created. |
| Provisioning | The *provisionRequest* has been sent or received and the Provisioning is in progress. |
| Provisioned | The Connection has been successfully provisioned |
| Activated |  |
| Releasing | A *releaseRequest* has been sent or received and a release of the connection is in progress |
| Terminating | a *terminateRequest* message has been received from the parent and waiting for connection termination by all children |
| Terminated | Nothing exists any longer, a terminate request has been successful. The current state-machine still receives messages from the children and passes it up to the parent |

Table : List of uRA and Aggregator states

### Ultimate PA State Machine

The uPA v2.0 state machine is represented below (see Figure 6). Several design decisions reflected in this state machine include:

* *In the “Auto Provisioning” state, a release request (“rel.rq”) transitions the state machine back to the “Reserved” state*. This state transitions allows the user to stop a circuit from being activated even when it is already in the “Auto Provision” state.
* *In the “Activating” state, an activation failure transitions the state machine to the “Scheduled” state*. In the event of an activation failure, the reservation is not immediately canceled but goes to the “Scheduled” state if the time window is within the reserved period of the reservation. This is done to facilitate troubleshooting for subsequent re-provisioning. This is particularly useful for long lived reservations, where re-requesting a large number of network resources may be inconvenient.

Figure 6. Ultimate Provider NSI CS “Oxford” State Machine

|  |  |
| --- | --- |
| **State** | **Description** |
| Initial (pseudo state) | the state-machine is not created until a *reserveRequest* is received |
| Reserving | A *reserveRequest* has either been sent or received and the processing is in progress. |
| Reserved | The *reserveRequest* has succeeded and a connection schedule has been created. |
| AutoProvision | A *provisionRequest* has been sent or received before the requested connection *startTime*. The NSA will now automatically provision the Connection at *startTime*. |
| Scheduled | The Reservation *startTime* has been reached and the Connection is ready to be provisioned as soon as the *provisionRequest* message is received. |
| Activating |  |
| Activated |  |
| Provisioning | The *provisionRequest* has been sent or received and the Provisioning is in progress. |
| Provisioned | The Connection has been successfully provisioned |
| Releasing | A *releaseRequest* has been sent or received and a release of the connection is in progress |
| Terminated | Nothing exists any longer, a terminate request has been successful. The current state-machine still receives messages from the children and passes it up to the parent |

Table : List of uPA states

## Requesting a Reservation

A *reserveRequest* is sent by the RA to the PA to initiate the lifecycle of the Connection. Once sent, both the RA and PA state-machines transition from Initial state to Reserving state. The PA reserves resources requested using path computation and depending on the results may choose to contact other NSAs to complete the reservation process. If the Reserving process completes successfully (also in all child NSAs) a *reserveConfirm* message is sent to the RA and the Connection moves into the Reserved state.

If the reservation process fails locally or in any of the child NSAs (due for example to the failure of path computation), then the PA issues a *reserveFailed* message to the RA, and the Connection moves to the Cleaning state in both the RA and PA and issues a *terminateRequest* to all child NSAs and a terminate instruction to the local NRM..

Once the local NSA responds with a *terminate\_ok* message and any child NSAs come back with a *terminateConfirm* message, the state-machine is terminated.

## Provisioning a Connection

When the Connection is in the Reserved state the RA can send a *provisionRequest* message. This request will be treated in two possible ways depending on the arrival of the request in relation to the startTime specified in the *reserveRequest* message:

* **Manual Start:** Where the *startTime* has already passed (according to the PA local timer), receipt of the *provisionRequest* message moves to Provisioning state.
* **Auto Start:** Were the *startTime* has not yet arrived (according to the PA local timer), the Connection moves to the *AutoProvision* state and waits until the *startTime* is reached. The Connection state then moves to Provisioning without further action by the RA at the commencement of the *startTime*.

When the local NRM or child NSAs indicate that the Provisioning has been completed, the PA issues a provisionConfirm message to the RA and the Connection moves to the Provisioned state. If the Provisioning fails, a *provisionFail* message is issued by the PA to the RA. No further action is taken – the Reservation moves to Scheduled state.

Connection lifecycle remains in the Scheduled state until a) a *provisionRequest* is re-tried, b) a *terminateRequest* is received, or the Reservation expires (end\_time is reached) and is automatically terminated by the PA.

A *provisionRequest* for a Connection already Provisioned is allowed and does not affect the service instance. A *provisionConfirmed* is returned. This action is specified in order to easily recover and synchronize connections that may have otherwise had piecewise interruptions to children.

## Releasing the Provisioning state

When a Connection is in the Provisioned state, the RA can send a *releaseRequest*. When a PA receives this request the Connection moves from the Provisioned state to the Releasing state and the local NRM and child NSAs are notified to de-provision the data plane resources associated with this connection. De-provision means that the data plane is no longer operational, but the resources remain reserved for the Connection in question. When the local NRM and all child NSAs indicate that the de-provisioning has been completed, the PA issues a *releaseConfirm* to the RA and the Connection moves to the Scheduled state.

If the local NRM or any child NSA fails to release completely, a *releaseFailed* message is issued to the RA and the connection moves to Scheduled state.

A *releaseRequest* for a connection already released is allowed but no action is performed.

## Terminating a Connection

In any state the RA may send a *terminateRequest* message to the PA. The Connection will then immediately move to the Terminating state, initiate a removal of the local Reservation and forward the *terminateRequest* to all child NSAs. Once the Reservation has been removed (and if necessary the Provisioning cancelled), both locally and on all child NSAs, the Connection moves to the Terminated state and the *terminateConfirm* message is sent to the RA.

## Forced end

The PA may force the end of a Reservation. In this case a *forcedEndRequest* message is sent upstream and the state changes to Cleaning state. In the cleaning state the RA removes the Reservation.

## Querying a Connection

The RA may send a query to the PA and the PA may send a query to the RA to find the state of a Connection. The PA returns the information about all service attributes associated with the connection as resolved in satisfying the *reserveRequest*. A query can request information about one or more Connections. Two query types may be requested, a summary or a detailed query. A summary query will return local information only, i.e it will not walk the NSA tree. A detailed request will instigate an attempt to walk the NSA tree and collect information about all of the children relating to the nominated Connection.

# NSI Framework Header

*(Note: should this be moved to the NSI framework document?)*

This section describes the new NSI framework header as implemented in NSI 2.0. The structural layout of the header is provided, along with definitions for each field, and a protocol compliance table outlining when fields must be included.

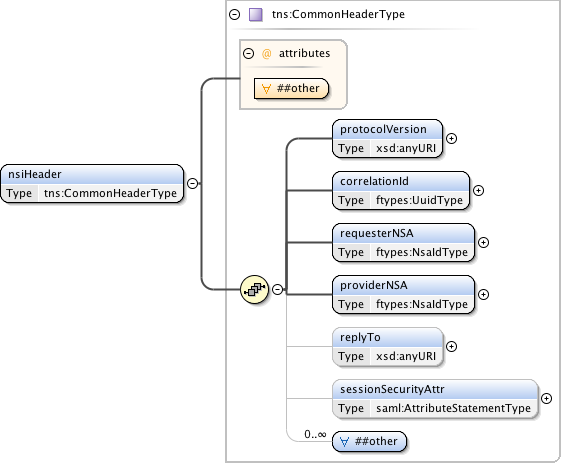


Figure – Common NSI 2.0 framework message header.

Figure 2 shows the new NSI 2.0 message header sent as part of every NSI message exchange. The original Oxford proposal was to create two separate header definitions, one for operation requests, and one for confirmed, failed, and notification messages. After further consideration a single header definition with optional elements was created to capture the semantics of both the request and response headers. The following attributes and elements are defined as part of the new NSI protocol header:

**Mandatory Elements:**

*protocolVersion*

A URI identifying the specific protocol version carried in this NSI message. The protocol version is modeled separately from the namespace of the WSDL and XML schema to capture behavioral changes that cannot be modeled in schema definition, and to avoid updating of the schema namespace.

*correlationId*

An identifier provided by the requesting NSA used to correlate to an asynchronous response from the responder. It is recommended that a Universally Unique Identifier (UUID) URN as per IETF RFC 4122 be used as a globally unique value.

*requesterNSA*

The NSA identifier for the NSA acting in the Requester Agent role for the specific NSI operation.

*providerNSA*

The NSA identifier for the NSA acting in the Provider Agent role for the specific NSI operation.

**Optional Elements:**  
  
*replyTo*

The Requester NSA's SOAP endpoint address to which asynchronous messages associated with this operation request will be delivered. This is only populated for the original operation request (i.e. reserve, provision, release, terminate, and query), and not for any additional messaging associated with the operation. If no endpoint value is provided in a operation request, then it is assumed the requester is not interested in a response and will use alternative mechanism to determine the result.

*sessionSecurityAttr*

Security attributes associated with the end user’s NSI session. This field can be used to perform authentication, authorization, and policy enforcement of end user requests. Is only provided in the operation request (i.e. reserve, provision, release, terminate, and query), and not for any additional messaging associated with the operation.

*other (any)*

Provides a flexible mechanism allowing additional elements in the protocol header for exchange between two-peered NSA. Use of this element field is beyond the current scope of this NSI specification, but may be used in the future to extend the existing protocol without requiring a schema change. Additionally, the field can be used between peered NSA to provide additional context not covered in the existing specification, however, this is left up to specific peering agreements.

**Optional Attributes:**  
*other (anyAttribute)*

Provides a flexible mechanism allowing additional attributes in the protocol header for exchange between two-peered NSA. Use of this attribute field is beyond the current scope of this NSI specification, but may be used in the future to extend the existing protocol without requiring a schema change. Additionally, the field can be used between peered NSA to provide additional context not covered in the existing specification, however, this is left up to specific peering agreements.

In addition, we identify the specific NSI CS operation being invoked using the “Soapaction:” element in the HTTP header as per section 6.1.1 of “Simple Object Access Protocol (SOAP) 1.1” found at <http://www.w3.org/TR/SOAP>. Discussion occurred in Oxford on the topic of including a specific “operation” element within the NSI header, however, this would have been a duplicate of the “Soapaction:” element, and therefore, was left out.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | *Header parameters* | | | | | | | |
| M = Mandatory  O = Optional  N/A = Not Applicable |  | *protocolVersion* | *correlationId* | *requesterNSA* | *providerNSA* | *replyTo* | *sessionSecurityAttr* | *other* | *Soapaction* |
|  | reserve | M | M | M | M | M | M | O | M |
|  | reserveConfirmed | M | M | M | M | N/A | N/A | O | M |
|  | reserveFailed | M | M | M | M | N/A | N/A | O | M |
|  | provision | M | M | M | M | M | M | O | M |
|  | provisionConfirmed | M | M | M | M | N/A | N/A | O | M |
|  | provisionFailed | M | M | M | M | N/A | N/A | O | M |
| Messaging  Primitives | release | M | M | M | M | M | M | O | M |
| releaseConfirmed | M | M | M | M | N/A | N/A | O | M |
| releaseFailed | M | M | M | M | N/A | N/A | O | M |
|  | terminate | M | M | M | M | M | M | O | M |
|  | terminateConfirmed | M | M | M | M | N/A | N/A | O | M |
|  | terminateFailed | M | M | M | M | N/A | N/A | O | M |
|  | query | M | M | M | M | M | M | O | M |
|  | queryConfirmed | M | M | M | M | N/A | N/A | O | M |
|  | queryFailed | M | M | M | M | N/A | N/A | O | M |
|  | forcedEnd | M | M | M | M | N/A | N/A | O | M |
|  | acknowledgment | M | M | M | M | N/A | N/A | O | M |

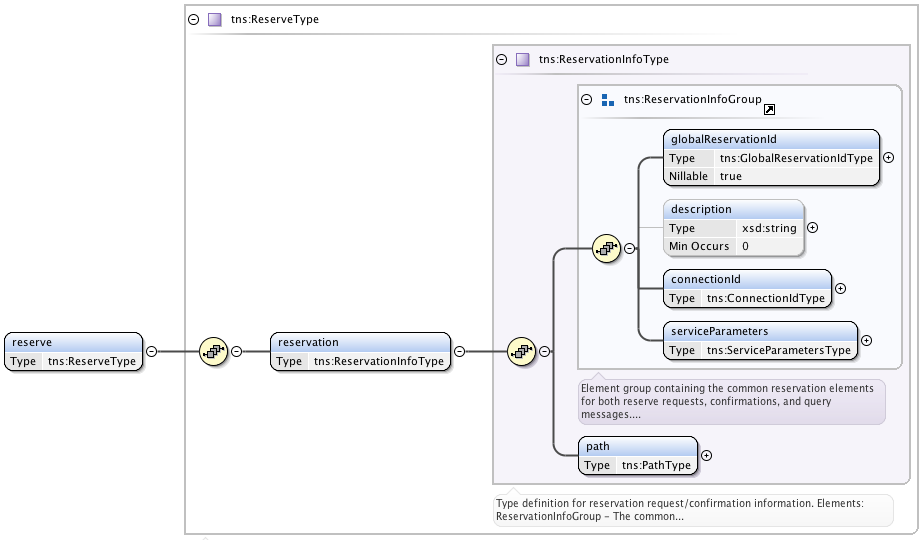
Table – NSI CS message use of header fields

# NSI Connection Services Primitives

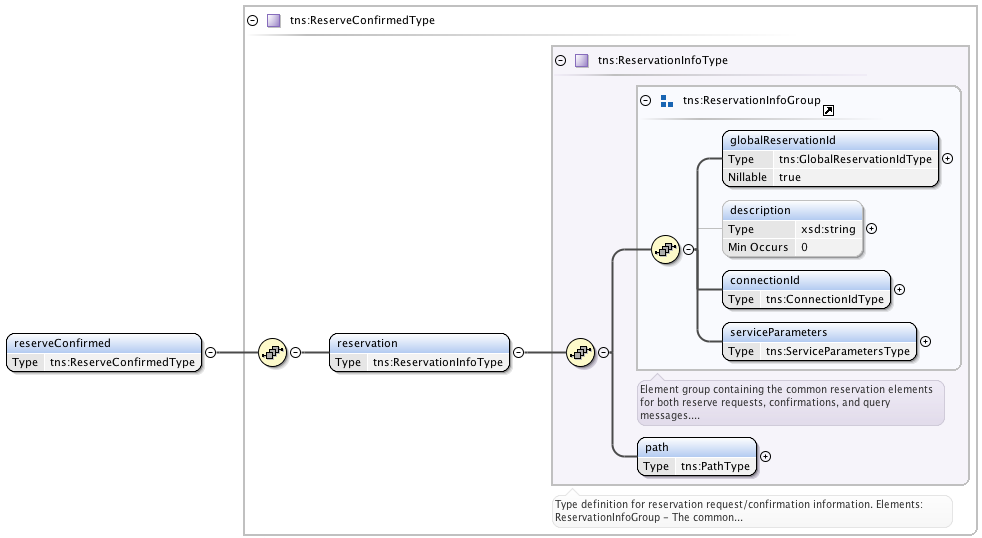
Restructuring of the NSI message layout into a common NSI messaging header and CS specific message body has resulted in a simplified definition of the NSI CS operation message structures. This section documents the new schema definitions for these primitives.

## Reserve

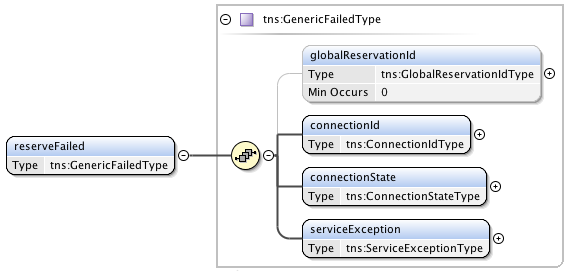
Make a request to reserve network resources for a connection between two STP's constrained by a certain service parameters.



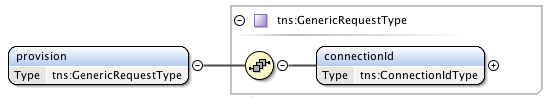
## ReserveConfirmed



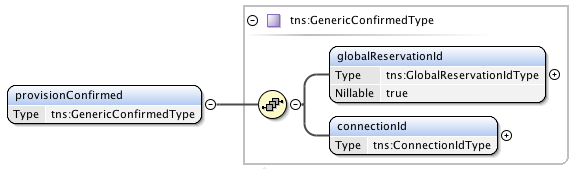
## ReserveFailed



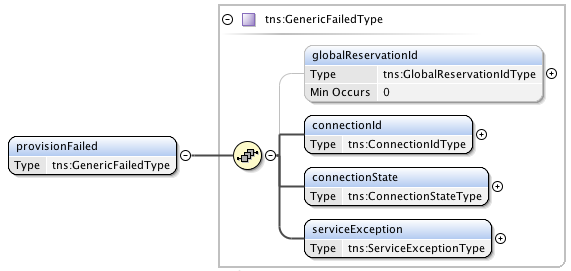
## Provision



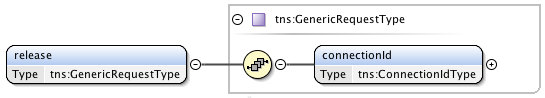
## ProvisionConfirmed



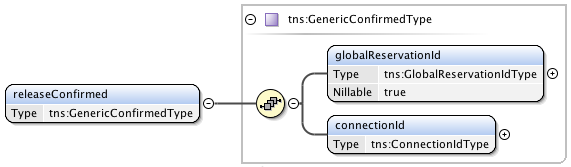
## ProvisionFailed



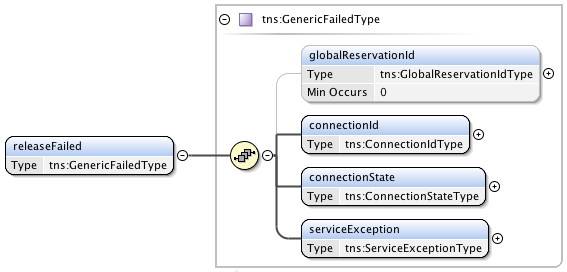
## Release



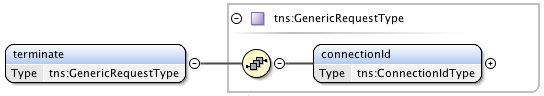
## ReleaseConfirmed



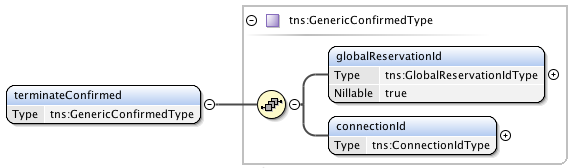
## ReleaseFailed



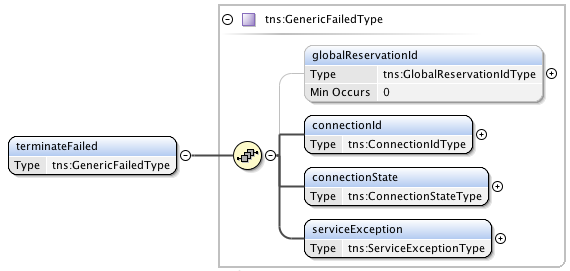
## Terminate



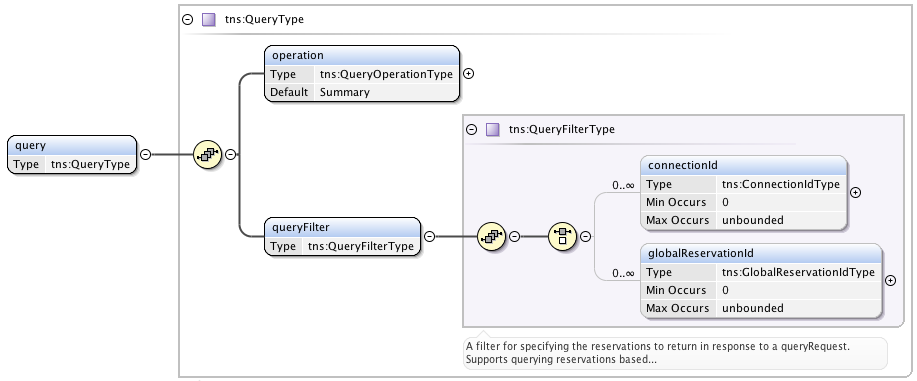
## TerminateConfirmed



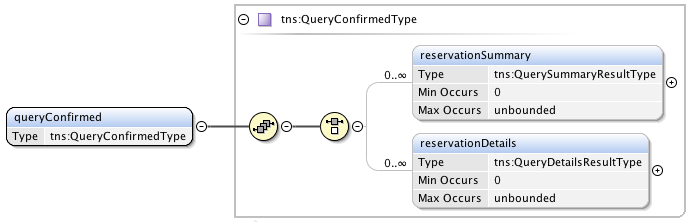
## TerminateFailed



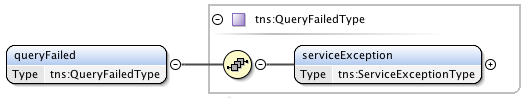
## Query



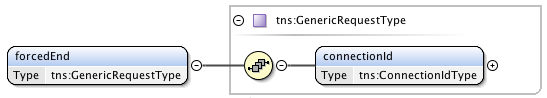
## QueryConfirmed



## QueryFailed



## ForcedEnd



# Connections: Transport and Service planes

The NSI Connection Service defines an abstracted representation of a Connection which is used in the service plane. This NSI Connection describes a conduit through which information is delivered from an ingress point to the egress point. While the model supports a uni-directional model of a connection, only point to point bidirectional symmetric Connections are supported in NSI version 1.0.

As illustrated in Figure 3, the Connection consists of three basic components: an ingress point where user data enters the connection, a transport section that carries the data across the network, and an egress point where user data exits the connection. These ingress and egress points are called Service Termination Points (STPs). This technology agnostic model of a Connection allows the NSI protocol to function with many different types of transport technologies.

The NSI CS protocol is an inter-network protocol, that is, it allows Connections to be built across multiple providers. The NSI v1.1 topology consists of STPs and Networks. Networks are groups of STPs configurable from a single NSA. Connections can be constructed between pairs of STPs and transit Networks - see Figure 3. In NSI v1.1 it is assumed that any two STP within a particular Network are able to be connected. This implies that all Networks are non-blocking.

Connections within a Network are intra-domain functions, and the technology details of how two STPs are actually connected is up to the local NRM and not a concern of the NSI protocols.

Two NSI Networks interconnect at a shared point known as an SDP. An SDP is a grouping of two adjacent STPs belonging to different Networks. A complete Connection can be built up by concatenating individual Connections at SDPs.

figure2

Figure : Inter-Network representation of a Connection

Once instantiated, an STP may have properties such as a framing type or a VLAN id. Some of these properties may reflect the options that can specified in the Service Definition. Labeling (cf. fiber id, wavelength, VLAN id) and aggregation (cf. combining multiple switch ports) can be modeled as a property of an STP.

The job of the physical instance of the Connection is to transport the user data (the “payload data”) across each section of the network inside a framing protocol. The framing protocol provides the necessary timing, control, and data integrity functions required to move the payload from node to node through the network. It is important to distinguish between a) the access framing protocols and b) the transport framing protocols.

It is the user payload data stream that is preserved from ingress to egress in a Connection. The transport framing can be any framing protocol as long as the end-to-end preservation requirement is honored. In fact, the only constraints on the transport framing is that the transport section be able to adapt the ingress payload data to each of the successive transport protocols that may be used along the path and ultimately be able to adapt the user payload to the egress framing at the egress point. While specification of the connection end points, access framing, and other parameters associated with a connection are defined by the connection requester (or implicitly by the service definition), the choice of the transport protocol and associated transport path parameters are explicitly delegated to the network service provider in order to allow the provider the greatest latitude in finding a valid, available, and optimal path for the connection request.

# Connection service use of the NSI Topology

As the name suggests the NSI Connection Service makes it possible to create connections. In order to do that some representation of the topology is required. In this section we describe what is necessary for the topology to support, what kind of elements should be in there and how it can be distributed.

## Representation of NSI Topology

In order to use the NSI, some form of topology representation is required. An introduction to this representation and the issues involved in creating network representations for the NSI is described below.

### Introduction to concept of STPs

The basic network topology for the Network Service Interface consists of points and connections. The NSI can be used to request a connection between two different points, which is then implemented using the connection(s) between those points. Since each of the points in the network topology can terminate a network service, they are called Service Termination Points (STPs). The STPs in topologies are identified using a unique identifier.

In practice, these STPs can represent ports on a switch, physical nodes, or even a specific VLANs on a port. This all depends on the local implementation.

The basic NSI topology is represented in a syntax that is explained in Appendix XX. This only provides the basic external topology. Any more detail can be added using the NML-WG topology representation.

### Requesting a path using STPs

To request a connection between two points in the network a client sends his request to an appropriate NSA using the reserve message. The exact details of the reserve message are discussed later, important here is that the request must contain a source STP and a destination STP. These define the endpoints for this connection.

The NSA will perform a pathfinding operation, and find a path between these two points. If it is an Aggregator NSA, it may divide this path into different segments, and using the same kind of reserve messages request the appropriate section in each of the networks involved.

### Under-specified STPs

The NSI v2.0 specification also allows clients to use under-specified STPs in their request. An under-specified STP is one which does not point to a specific point in the topology, but rather points to a group of candidates. An under-specified STP is always part of a domain, and the NSA local to that under-specified STP will make a decision on which specific STP to use in place of the under-specified STP.

The reply back to the requester will contain the fully specified STP selected for the request.

An example for this kind of request is by specifying an STP which has VLAN labels, but not requesting a specific VLAN label. The NSA will then select an available label at its discretion.

### Explicit routing using STPs

In a regular request only the source and destination STPs are specified. The selected path between those STPs is left to the NSA managing the request. In NSI v2.0 it is also possible to steer the requested path into a specific direction by defining intermediate STPs that the path must touch.

* Support for unidirectional connections
* Support for STP type value pairs
* Support for bunldes of STP type-value pairs
* Common mapping mechanism to NML Ports
* The data model should be independent of the underlying technology
* Advertise also domain-internal connectivity using NML constructs

## STPs and Paths

### STP syntax

The STP topology uses STPs to identify network resources, where the *stpId* is a tuple of *NSnetwork:*<localid>.

*Where, NSnetwork* = urn:ogf:network:nsnetwork:<networkId>.

And<localid> = A locally unique identifier for the STP within the target network.

### Under-specified STPs

\*\*\* needs updating

### Path

The *Path* describes a route through the topology. When present in a Connection Request, the Path specifies a set of Service Termination Points (STPs) that the Connection transits, and in the order the connection must transit them. Within a *reserveRequest*, the *Path*, at a minimum, must specify the ingress and egress STPs for the Connection. Additional intermediate transit points may be included, and when present, they are considered a required constraint on the Connection’s route and must be honored.

The *Path* is made up of the following attributes: *directionality, sourceSTP/ destSTP, stpId, stpSpecAttrs and stpList* as described earlier in this document. The *stpList* is a simple ordered list if list of Service Termination Points (STPs).

### Path example in an NSA chain

An example of a *Path* in a NSA chain is presented to show how a *Path* can be described.

path_chain

Figure 4: Example of Connection managed by a NSA chain

In this example there is an Inter-Network Topology consisting of 3 networks, one per NSA. Each Network is described as a set of edge points on a network.

For this example the topology would look like this:

Network X: X:a X:b

Network Y: Y:c, Y:d, Y:e

Network Z: Z:f, Z:g

In this example, the NSAs are connected as a chain:

NSA-X(Requester) to NSA-Y(Provider), NSA-Y(Requester), to NSA-Z(Provider)

Assuming a Request comes from the Application-NSA to NSA-X to reserve a connection X:a to Z:g, then NSA-X will look in the topology and determine that to make this Connection, NSA-X will reserve a local connection from X:a to Xb, and then NSA-X must forward a request for the remainder of the connection to NSA-Y: Y:C to Z:g

NSA-Y gets this request and reserves a connection between Y:c and Y:e and requests a Reservation from NSA-Z for a connection Z:f to Z:g.

### Path example in an NSA tree

An example of a *Path* in a NSA tree is presented to show how a *Path* can be described.

path_tree

Figure 5: Example of a Connection managed by a NSA tree

The topology remains the same as for the previous example:

Network X: X:a X:b

Network Y: Y:c, Y:d, Y:e

Network Z: Z:f, Z:g

In this example, the NSAs are connected as a tree:

NSA-X(Requester) to NSA-Y(Provider) and

NSA-X(Requester) to NSA-Z(Provider)

Assuming a request comes from the Application-NSA to NSA-X to reserve a connection X:a to Z:g, then NSA-X will look in the topology and determine that to make this connection, a connection request is required locally between X:a and X:b. Next NSA-X must forward two requests:

1. To NSA-Y: Y:c to Y:e
2. To NSA-Z: Z:f to Z:g

## Inter-domain vs intra-domain pathfinding

There are two levels of pathfinding related to the NSI architecture: the inter-domain pathfinding which is concerned with choosing a path across the global topology of Networks, and the intra-domain pathfinding concerned with the transport resources within the Network. NSI is concerned only with inter-Network pathfinding.

Inter-Network Connections extend across multiple networks; they are constructed by concatenating connections built across the individual networks. This is done by choosing appropriate STPs such that the egress STP of one connection corresponds directly with the ingress STP of the successive connection.

Both the Tree and Chain model reduce pathfinding to a constraint-based search over a topology to build a k-preferred path tree. The method, tree or chain, used to process a request is made exclusively in the requester NSA. No path finding algorithms (e.g. PCE or OSPF-TE) are mandated by NSI.

## Distribution of NSI Topology

Some form of Topology distribution is required in order for an inter-domain NSI network to function. In NSI 1.0 this process was performed out-of-band, mostly through e-mail. For NSI 2.0 we take the opportunity to provide a standardized interface for NSI topology exchange.

### Transport and Service plane relationship

The NSI Connection Service is implemented on Network Service Agents (NSAs), which together form a network and tree-like structure. This graph represents how reservation requests would propagate through the network, but not necessarily reflects the transport-plane. One NSA may be an aggregation point for other NSAs, not visible from the outside.

The messaging between the NSAs will happen on the service plane, which is completely separate from the transport plane.

### Bootstrapping Topology Exchange

To start the initial Network Service the NSAs must be able to find each other, in order to communicate details about the network. So some form of bootstrapping is required, with initial synchronization between domains on both the service plane and the transport plane, i.e. the NSAs of both domains must be able to contact each other, and the details of transport plane connections between them have to be synchronized as well.

Initiating a transport plane connection between two networks is not a frequent occurrence, and a longer process, involving out-of-band (for NSI) contact. Part of that process can be that the networks also communicate the access details for the NSAs, thus forming an NSA relationship.

### Expanding the Topology Exchange

Once the neighboring NSAs have exchanged details, they can also distribute details about the rest of the network, both the control plane details and connectivity, but also some transport details. This process can be similar to the OSPF bootstrap procedure.

### Update Mechanism

The transport network is not static, and links are added or removed from time to time. An update mechanism is thus required to inform other NSAs about these kinds of events. In OSPF these kinds of updates are sent using multicast, which is not always available over the Internet. Another option is to form a peer-to-peer network between the NSAs and send updates over this network.

### Policy on Topology Distribution

Some networks are willing to provide additional details on their internal network to others, sometimes on limited conditions.

For NSI version 2.0 we acknowledge this issue, but provide no standard mechanism. We have not seen clear requirements or specific use-cases for these mechanisms to provide a sensible protocol for it.

# Security

• Inder, Henrik and John M.

• 25 April 2012: no updates on progress

•

## Authentication and Authorization

### Security Requirements

The basic security requirements of any trusted, distributed service are: 1. The requester and provider can be mutually authenticated: 2. The messages between them cannot be secretly tampered with (message integrity), 3. The provider should be able to get enough trusted information about the requester to satisfy its authorization requirements.

### Message Security

Message integrity and authentication should be provided by the transport protocol over which the NSI messages are sent. Some examples of underlying secure protocols are: a VPN, Transport Layer Security (TLS), or SOAP with digitally signed messages. The choice of this protocol is not included in this specification, but should be addressed in a Security Profile (see the NSI CSv1 Security document for an example).

### Authorization

Authorization of resource use may be based on bilateral trust agreements between an RA and a PA. In addition, the NSI connection protocol provides a means of carrying identifying information on which authorization can be based. All NSI messages contain *securityAttr* objects associated with the requesterNSA and the providerNSA. The *securityAttr* object is left opaque in the schema to facilitate various types of authorization schemes. One model that can be supported is to provide a set of mutually agreed upon attributes for the requester, The NSI CS v1.1 document specifies a schema for an *securityAttr* that contains set of defined attributes that can apply to the RA or to the user who originated the request.

## Failures and exceptions

### Service plane failures

Service plane failures are detected by the NSI transport and as such is not part of this NSI CS protocol. See Appendix A: Best Practices for NSA implementation for a discussion on handling time-outs.

### Transport plane failures

Failures in the transport plane can occur at any time, however within the framework of the NSI architecture, there are two time windows in which a transport plane failure is significant:

1. The time between the service Reservation completed and Provisioning start
2. The time between the service Provisioning completed and teardown started

The errors only need to be handled by the NSA if the Data Plane errors affect the user service.

Figure : Local/Remote Failures

Transport failure during the service Reservation and Provisioning: An element in the Data Plane becomes unavailable due to a soft or hard failure causing a Provisioning failure of a confirmed Reservation; the NRM can handle this by either reserving an alternate path as long as it meets the requested service characteristics or applying a forcedEnd to the Reservation. The local policy of a Network provider and availability of resources will determine what recovery action is taken.

Transport failure during Provisioning phase and teardown phase: In case a failure in the Data Plane affects an active Connection, the first recovery mechanisms will be triggered by the protection mechanisms Provisioned with the service. If the Connection Service is unprotected, then the failure notification will be sent to the Domain’s NSA. At that point, NSA will take appropriate action based on service and user policies by either re-routing the Connection within the Network or tearing down the service.

# Appendix A: Best Practices for NSA implementation

This appendix lists a set of best practices to ensure interoperability between NSA implementations.

## MDL

The following (see Figure 4) is an example workflow of the MDL.

Figure 4. Example workflow of MDL

1. NSA 1 sends messages to NSA 2 and NSA 3 via the MDL.
2. NSA 2 and NSA 3 acknowledge receipt of the message.
3. The MDL in NSA 1 flags that both children (NSA 2 and NSA 3) received the message successfully.
4. NSA 3 sends messages to NSA 4 and NSA 5 via the MDL.
5. NSA 4 acknowledges receipt (NSA 5 does not).
6. The MDL in NSA 3 recognizes that NSA 5 did not acknowledge receipt of the message and retries sending the message.
7. NSA 3 resends the message to NSA 5 via the MDL (and may continue to do so for multiple times if it does not get an acknowledgement).
8. The MDL in NSA 3 flags that it did not get an acknowledgement from NSA 5 (after multiple attempts), gives up, and reports a fatal error.

## Message transport error handling

Additional error condition handling: The following set of checks is required to pass for messages to be considered vaild, otherwise a message transport layer fault will be returned:

* HTTP authentication – if the message does not have valid credentials it will be rejected with an HTTP 40x message.
* *correlationId* - needed for any acknowledgment, confirmation, or failed message. Must be unique within the context of the providerNSA otherwise the request cannot be accepted.
* *replyTo* - we will send the confirmation, or failed message back to this location. We do not validate the contents of the endpoint, just that it exists.
* *Reservation* – if the reservation parameters are not present then we reject.
* *requesterNSA* and *providerNSA* – must be present and resolve to an *NSnetwork* in topology. Also, the *providerNSA* must be the *NSnetwork* that the NSA is managing or the message will be rejected.
* *connectionId* – this is used as the primary reference attribute for Reservation state machines and must be present.
* If any of these fields are missing or invalid the NSA will return a message transport fault containing the *NSIServiceException* set to an appropriate error message. Typically this will be MISSING\_PARAMETER - "SVC0001", "Invalid or missing parameter" for this generic case and specify attributes identifying the parameter in question.

The following list of parameters should be validated when receiving a reservation message:

|  |  |  |  |
| --- | --- | --- | --- |
| *errorId* | *errorDescription* | *text* | *variables* |
| SVC0001 | MISSING\_PARAMETER | Invalid or missing parameter |  |
| SVC0002 | UNSUPPORTED\_OPTION | Parameter provided contains an unsupported value which MUST be processed |  |
| SVC0003 | ALREADY\_EXISTS | Schedule already exists for connectionId |  |
| SVC0004 | DOES\_NOT\_EXIST | Schedule does not exist for connectionId |  |
| SVC0005 | MISSING\_SECURITY | Invalid or missing user credentials |  |
| SVC0006 | TOPOLOGY\_RESOLUTION\_STP | Could not resolve STP in Topology database |  |
| SVC0007 | TOPOLOGY\_RESOLUTION\_STP\_NSA | Could not resolve STP to managing NSA |  |
| SVC0008 | PATH\_COMPUTATION\_NO\_PATH | Path computation failed to resolve route for reservation |  |
| SVC0009 | INVALID\_STATE | Connection state machine is in invalid state for received message |  |
| SVC0010 | INTERNAL\_ERROR | An internal error has caused a message processing failure |  |
| SVC0011 | INTERNAL\_NRM\_ERROR | An internal NRM error has caused a message processing failure |  |
| SVC0012 | STP\_ALREADY\_IN\_USE | Specified STP already in use |  |
| SVC0012 | BANDWIDTH\_NOT\_AVAILABLE | Insufficient bandwidth available for reservation |  |

Table : error messages

\*\*\*We will also need to agree on the format of the message/errorId.

## ACK handling

Delays on the transport layer can result in ACK arriving after the confirm/fail message. The following guidelines are recommended for handling web-service ACKs:

1. For protocol robustness, the NSA should accept any confirm/fail messages even if these are received out-of-order w.r.t. the ACK, i.e. before the associate ACK has been received.
2. The receipt of a confirm/fail message cancels out the need to receive an ACK. So the NSA should not only continue to process the confirm/fail message, but not gate on or wait for the ACK, i.e consequent-messages may be sent without waiting on the receipt of the ACK.
3. The NSA should send the ACK before sending the associated confirm/fail message.
4. The message transport layer takes care of ACK retransmission in case of a packet loss.
5. If the message transport layer is broken, the ACKs will eventually timeout and generate a message transport error that the NSA will need to handle.

## Guidelines on timeouts:

1. Timeouts should be configurable on a per operation basis and set to 2 minutes as a default.
2. Requester side timeouts: It is up to the individual provider to choose appropriate NSA timeouts for their network. As a guide the timeout should be set to 1 minute for reservations to a provider only NSA, and longer for hierarchical requests to aggregator NSAs depending on the number of levels of recursion. Provisioning requests are likely to take longer than Reservation requests. The timeout will need to be tailored to meet the response times of the participating networks.
3. The requester NSA may choose to send queries to check the status of a request rather than terminating at timeout.

## Parallel processing of messages:

The provider NSA should respond to queries even if still working on a response to a request.

## NTP servers

The server running the NSA should use NTP version 4 [8]. This will reduce the risk of clock skew between the NSAs.

# Appendix B: Service Definitions for Connection Services

• Jerry Sobieski (notified)

• 25 April 2012: no updates on progress

• Service definitions

# Appendix C: NSI topology

## STP, SDP, Networks, NSAs concepts and syntax

Version 1 of the NSI Connection Service specification left the topology definition out of scope. This has left a huge gap for the operational side, where implementers have had to cooperate to create a common file to represent the topologies of each of the domains, and how to share that data.

The Distributed TOpology eXchange (DTOX) working group of GLIF jumped to the opportunity and quickly provided a topology format which was heavily inspired by the NML work in progress, but also contained some additions specific to NSI. This has allowed us to gain some experience in required elements for a topology format, and the way it could be exchanged.

The current DTOX format contains the following elements:

* **STP** Service Termination Point.
  + **connectedTo** relation to form an SDP with another STP
* **NSNetwork** Network Service Network
  + **hasSTP** to define STP containment
  + **locatedAt** to define a location of a network
* **Location**
  + **lat, long** define GPS coordinates
* **NSA** Network Service Agent
  + **managing** to relate it to an **NSNetwork**
  + **adminContact** to describe contacts for the administrator
  + **csProviderEndpoint** to define the URL at which the NSA is reachable

The above format is simple, but effective. The STP elements are the most important ones which provide the identifiers and connectivity information necessary to do path calculations between domains. An example is provided below. This uses the DTOX namespace, for the final NSI v2.0 version this would be replaced with the NSI concepts and an NSI namespace.

<rdf:RDF xmlns="http://www.glif.is/working-groups/tech/dtox#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:dtox="http://www.glif.is/working-groups/tech/dtox#">

<owl:NamedIndividual rdf:about="urn:ogf:network:stp:netherlight.ets:uva-83">

<rdf:type rdf:resource="http://www.glif.is/working-groups/tech/dtox#STP"/>

<dtox:connectedTo rdf:resource="urn:ogf:network:stp:uvalight.ets:ams-83"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="urn:ogf:network:nsnetwork:netherlight.ets">

<dtox:hasSTP rdf:resource="urn:ogf:network:stp:netherlight.ets:uva-83"/>

<dtox:managedBy rdf:resource="urn:ogf:network:nsa:netherlight"/>

<dtox:locatedAt rdf:resource="urn:ogf:network:nsnetwork:netherlight.ets:location"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="urn:ogf:network:nsnetwork:netherlight.ets:location">

<rdf:type rdf:resource="http://www.glif.is/working-groups/tech/dtox#Location"/>

<dtox:lat rdf:datatype="http://www.w3.org/2001/XMLSchema#float">52.357</dtox:lat>

<dtox:long rdf:datatype="http://www.w3.org/2001/XMLSchema#float">4.953</dtox:long>

</owl:NamedIndividual>

<!-- urn:ogf:network:nsa:netherlight -->

<owl:NamedIndividual rdf:about="urn:ogf:network:nsa:netherlight">

<rdf:type rdf:resource="http://www.glif.is/working-groups/tech/dtox#NSA"/>

<rdfs:label xml:lang="en">netherlight</rdfs:label>

<dtox:managing rdf:resource="urn:ogf:network:nsnetwork:netherlight.ets" />

<dtox:adminContact rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

OpenDRAC (SURFnet)

Proj Manager: John MacAuley

email: john.macauley@surfnet.nl

phone: 1-613-220-6817

skype: john.macauley

Software Developer: John MacAuley

email: john.macauley@surfnet.nl

phone: 1-613-220-6817

skype: john.macauley

</dtox:adminContact>

<dtox:csProviderEndpoint rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

http://dracproxy01.surfnet.nl:8080/nsi-v1/ConnectionServiceProvider

</dtox:csProviderEndpoint>

</owl:NamedIndividual>

</rdf:RDF>

The NSNetwork element defines how the STP elements are connected internally, i.e. full mesh connections. Note that so far we have only seen a one-to-one mapping of NSNetworks to NSAs.

The NSA provides the management information for networks, how the NSI interface can be reached, and who actually maintain the NSA.

The Location element has also proven to be quite useful in allowing us to quickly create stunning visualizations using Google Earth.

### Possible Improvements

At the moment all of the NSAs can be reached over the regular Internet, and there is no policy in place restricting access to them. In the future it may be necessary to also describe the service plane connectivity. We would need to extend the description format to also describe that accurately.

## Relationship between NSI topology and NML

The NSI Topology in the DTOX format described above currently does not describe internal details of a domain’s connectivity. A fully-connected mesh is assumed between all the ports connected to an NSNetwork, and it is not possible to provide additional details about the internal connectivity, such as delay, jitter, and other quality of service attributes, or even that there is no connectivity at all.

With NSI v2.0 it is possible to provide additional details on the internal topology if so desired. The format for this is the NML OWL syntax. The identifier of an STP is then reused to define that it also is an NML Port. Using that as a starting point it becomes possible to relate it to other NML objects such as other Ports, Links, Nodes, Domains, et cetera. For more details see the NML specification.

## NSI topology best practices

• Chin and Inder

• 25 April 2012: no updates on progress

• Best practices for pathfinding using NSI topology

• Best practices for topology exchange

# Appendix D: Information Services

• John M (notified)

• 25 April 2012: John will tackle after WSDL completing updates

• NSI version number discovery

• Service Definition discovery

# Appendix E: Service Definitions for Connection Services

The NSI “Service Definition” defines the characteristics that a particular Connection Service will honor when establishing Connections.

The Service Definition takes two forms: The first is a human readable document that describes the nature of a particular NSI Service. This human oriented document describes the service capabilities in a comprehensive fashion for human consumption and serves as a basis for the second form of SD which is a machine readable XML based web document that allows automated agents to learn and validate NSI service attributes.

The SD for people describes the intent of the service – what it is meant to provide. The document includes the technical parameters and capabilities that the service will support. This human readable version also describes wide range of other aspects of the services such as how the service will be supported by Network Operations Center(s), response time for problems, how the service is to be monitored for either long term accounting or short term problem resolution, etc.

The machine readable form for the Service Definition is initially meant to provide a means for automated agents to learn which specific connection attributes are recognized by a particular service domain. This web based XML document should be in a “well known” location such that automated agents broadly speaking can find and fetch the XML SD.

An example machine form Service Defintion:

<ServiceDefinition>

<ServiceTypeIdentifier> P2PCS </ServiceTypeIdentifier>

<AttributeList>

<Attribute>

<AttrName> Capacity</AttrName>

<AttrDesc>

Bandwidth capacity of the connection. Measured in bits/sec.

</AttrDesc>

<AttrType>Integer</AttrType>

<AttrFloor>0</AttrFloor>

<AttrCeiling>100 000 000 000</AttrCeiling>

</Attribute>

<Attribute>

<AttrName>MBS</AttrName>

<AttrDesc>

Maximum burst size. Measures in bits. This is the mximum amount of contiguous data that can be presented at ingress before the source must pause. The pause must last long enough to realize the appropriate duty cycle for that STP and connection capcty.

</AttrDesc>

<AttrType>Integer</AttrType>

<AttrFloor>10 000</AttrFloor>

<AttrCeiling>100 000</AttrCeiling>

</Attribute>

</ServiceDefinition>

There are two types of formal Service Definitions: A “Common” Service Definition (CSD) that all conforming networks will use as a basis for service capabilities, and a “Local” Service Definition (LSD) that describes local difference. The Common SD (CSD) describes the basic set of paramteres that all service implementations must recognize. The local SD (LSD) will describe essentially minor differences in capabilities of the local service offering from the ideals described in the common service definition.

For instance, the Common Service Definition for a point to point Connection Service (“P2PCS”) might define “Capacity = 0 to 100\*10^9 bps” – a 100Gbps capable service. But some local network only currently can do 10 Gbps maximum. That local service would offer a Local SD that overrides the Common SD. The”capacity” attribute must be recognized in the LSD, but the maximum would be specified as 10 Gbps (rather than 100 Gbps).

All services that implement a Common Service must recognize all of the Common SD attributes and treat them semantically the same. The ranges of those attributes may be adjusted by local SD specifications, but the CSD attributes will apply to all conforming Services.

How the SD works:

Service Definitions serve three basic purposes: 1) They provide a objective set of service specifications that can be implemented independently in different networks; 2) since NSI is technology agnostic, the CSD provides a means of objectively insuring that a correct and complete set of connection parameters were specified with respect to the service being requested.; and 3) path finders can formally interpret service parameters in order to prune large search spaces quickly and early.

The SDs are processed as follows: The Reservation Request specifies the type of service desired, for instance ServiceTypeIedentifier=”P2PCS”. The Provider Agent can fetch the CSD document corresponding to this Service Type and use that document to qualify the initial request. This qualification step basically insures that all user specified attributes are present, and defaults are applied where appropriate, This results in a “fully specified” request in which all constraints for a particular service have been verified.

The provider agent may use any appropriate path selection algorithm that meets the constraints in the reservation request. One possible such approach might be to examine each service domain to determine if it offers the requested Service Type. A domain announcing the desired Service Type thus becomes is a primary candidate for a path analysis. A group of inter-connected domains that all offer the same Service Type will form a single contiguous service region. In general, Connections across individual serove domains in such a region may be concatenated to form end to end connections. This regional homogeneity simplifies path finding substantially. (Indeed this is a useful trait of most such Connection Services. To be sure, this path selection process is not mandated, but understanding the nature of viable transit service domains can improve global path selection.

All networks that offer a given Service Type, will by definition, recognize and interpret all the attributes defined in the machine SD. A network may adjust the values of those attributes to reflect local limitations or enhancements, but they must receognize all the CSD attributes.

A network LSD may also recognize additional parameters that are not explicitly in the CSD. However, these local parameters must be optional. A required parameter in an LSD that does not appear in the CSD constitute s different service type.

# Appendix F: Changes to NSI state machine between v1.1 and v2.0

This appendix documents changes in the state machine compared to NSI CS v2.0 compared to v1.1

Four core changes have been made to the state machine:

1. V2.0 adds Activated state to provide more information in the case of a large time gap between >prov.rq and <prov.cf
2. V2.0 Allows unsolicited messages from the PA to the RA as a form of notification
3. V2.0 Formalizes a Message Delivery Layer (MDL)
4. V2.0 Separates the NSI CS v1.0SC state machine into two distinct state machines, one for the ultimate RA and Aggregator, and one for the ultimate provider.

## Activated state

The NSI CS v1.1 state machine required that **both a provision request message and the start time** trigger are received before initiating the initial setup of a circuit. A provision confirm message is not returned until the circuit has been successfully setup. In the current state machine, if the provision request message is sent well in advance (e.g. 1 week) of the start time, there can be a substantial amount of elapsed time before the provision confirm message is returned. To manage this eventuality the Activate state has been introduced.

## Unsolicited Messages from PA to RA (Notify)

The use of unsolicited messages from the PA to the RA (or notification) was devised primarily to communicate to the RA a local event in the PA that resulted in a state transition in the state machine. An example of this is the “activate\_ok.nt” and “activate\_ng.nt” notify messages (see Figure 4 and 5) sent from the PA to the RA to indicate a success or failure of the circuit setup in the PA.

## Message Delivery Layer (MDL)

The need for a new conceptual layer in the NSI protocol was identified during the evaluation of NSI v1.1. There is a need to confirm the delivery of messages to all immediate children including the ultimate PA. The MDL layer provides this function by adding mechanisms such as timeouts and retries. The MDL layer also aggregates the replies from multiple child NSAs.

## Separation of uRA/Aggregator, and uPA State Machines (SM)

The motivation for separating the NSI CS v1.0SC state machine into two distinct state machines was done primarily for correctness. In particular, it was to address the issue of aggregator NSAs transitioning to the “Provisioned” state without knowing if the connection in its children NSAs were active. In addition, the notion of the control plane provisioning was decoupled from the data plane circuit setup by the introduction of the “Activating” and “Activated” states. In the NSI CS v1.0SC state machine, the “Provisioned” state indicated that the circuit was setup and ready for data to flow over it. In the v2.0Oxford state machine, the “Provisioned” state simply indicates that the provision confirm message (i.e. “prov.cf”) has been received in acknowledgement of the provision request message (i.e. “prov.rq”). This represents the control plane workflow process. When the circuit is setup, a notification message (i.e. “activate\_ok.nt”) is sent from the PA to the RA to indicate and trigger a state change to the “Activated” state respectively. This represents the data plane workflow process. The table below (see Table 1.) is a summarization of the “Provision” and “Activated” states within the v1.0SC and v2.0Oxford state machines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | NSI CS v1.0SC | | NSI CS v2.0Oxford | |
| Provisioned | Control: | The provision request message (“*prov.rq*”) has been processed and acknowledged with a provision confirm message (“*prov.cf*”) | Control: | The provision request message (“*prov.rq*”) has been processed and acknowledged with a provision confirm message (“*prov.cf*”) |
| Data: | **Circuit is ready for data movement** | Data: | *No action taken* |
| Activated | Control: | *Not applicable* | Control: | An activate notification (“activate\_ok.nt”) was received |
| Data: | *Not applicable* | Data: | **Circuit is ready for data movement** |

Table 1. Comparison of “Provisioned” state between v1.0SC and v2.0 Oxford SM

# Appendix G: SOAP Messaging Examples

The following SOAP over HTTP message shows the new structural layout of the Reserve request primitive using data from the existing Automated GOLE demo system. This XML was generated using the Oracle (SUN) Metro SOAP stack with JAX-WS.

POST /nsi-v2/ConnectionServiceProvider HTTP/1.1  
Content-type: text/xml;charset="utf-8"  
Authorization: Basic bnNpZGVtbzpSaW9QbHVnLUZlc3QyMDExIQ==  
Soapaction: "http://schemas.ogf.org/nsi/2012/03/connection/service/reserve"  
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2  
User-Agent: JAX-WS RI 2.1.6 in JDK 6  
Host: localhost:8084  
Connection: keep-alive  
Content-Length: 2511  
  
  
<?xml version="1.0" ?>  
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
 <S:Header>  
 <ns5:nsiHeader xmlns:ns2="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/framework/headers"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/connection/types">  
 <protocolVersion>http://schemas.ogf.org/nsi/2012/03/connection</protocolVersion>  
 <correlationId>urn:uuid:b0dea3cf-dd4b-4c2c-bee3-f9ec8cf3affa</correlationId>  
 <replyTo>http://localhost:9080/nsi-v2/ConnectionServiceRequester</replyTo>  
 <requesterNSA>urn:ogf:network:nsa:netherlight</requesterNSA>  
 <providerNSA>urn:ogf:network:nsa:czechlight</providerNSA>  
 <sessionSecurityAttr>  
 <ns4:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" Name="globalUserName">  
 <ns4:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string"  
 >jrv@internet2.edu</ns4:AttributeValue>  
 </ns4:Attribute>  
 <ns4:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" Name="role">  
 <ns4:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string"  
 >AuthorizedUser</ns4:AttributeValue>  
 </ns4:Attribute>  
 </sessionSecurityAttr>  
 </ns5:nsiHeader>  
 </S:Header>  
 <S:Body>  
 <ns7:reserve xmlns:ns2="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/framework/headers"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/connection/types">  
 <reservation>  
 <globalReservationId>urn:ogf:network:service:9cd41aa0-b019-435f-b3dc-075098816737</globalReservationId>  
 <description>This is a test schedule connecting ps-80 to ams-80</description>  
 <connectionId>urn:uuid:deb6aa8d-3e59-4821-9f7f-5c808be42bde</connectionId>  
 <serviceParameters>  
 <schedule>  
 <startTime>2012-05-02T15:47:33.937Z</startTime>  
 <endTime>2012-05-02T15:49:33.945Z</endTime>  
 </schedule>  
 <bandwidth>500</bandwidth>  
 </serviceParameters>  
 <path>  
 <directionality>Bidirectional</directionality>  
 <sourceSTP>  
 <stpId>urn:ogf:network:stp:czechlight.ets:ps-80</stpId>  
 </sourceSTP>  
 <destSTP>  
 <stpId>urn:ogf:network:stp:czechlight.ets:ams-80</stpId>  
 </destSTP>  
 </path>  
 </reservation>  
 </ns7:reserve>  
 </S:Body>  
</S:Envelope>

The following message is the HTTP response containing the acknowledgement to the previous Reserve request.

HTTP/1.1 200 OK  
Server: Apache-Coyote/1.1  
Content-Type: text/xml;charset=utf-8  
Transfer-Encoding: chunked  
Date: Wed, 02 May 2012 15:45:34 GMT  
  
  
<?xml version='1.0' encoding='UTF-8'?>  
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
 <S:Header>  
 <ns7:nsiHeader xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns4="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/types"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/framework/headers">  
 <protocolVersion>http://schemas.ogf.org/nsi/2012/03/connection</protocolVersion>  
 <correlationId>urn:uuid:b0dea3cf-dd4b-4c2c-bee3-f9ec8cf3affa</correlationId>  
 <requesterNSA>urn:ogf:network:nsa:netherlight</requesterNSA>  
 <providerNSA>urn:ogf:network:nsa:czechlight</providerNSA>  
 </ns7:nsiHeader>  
 </S:Header>  
 <S:Body>  
 <ns6:acknowledgment xmlns:ns2="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns4="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/types"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/framework/headers" />  
 </S:Body>  
</S:Envelope>

The following SOAP over HTTP message shows the new structural layout of the ReserveConfirmed primitive in response to a successful schedule creation as requested in the previous Reserve request operation.

POST /nsi-v2/ConnectionServiceRequester HTTP/1.1

Content-type: text/xml;charset="utf-8"

Authorization: Basic bnNpZGVtbzpSaW9QbHVnLUZlc3QyMDExIQ==

Soapaction: "http://schemas.ogf.org/nsi/2012/03/connection/service/reserveConfirmed"

Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2

User-Agent: JAX-WS RI 2.1.6 in JDK 6

Host: localhost:9080

Connection: keep-alive

Content-Length: 2529

<?xml version="1.0" ?>  
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
 <S:Header>  
 <ns7:nsiHeader xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/types"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/framework/headers">  
 <protocolVersion>http://schemas.ogf.org/nsi/2012/03/connection</protocolVersion>  
 <correlationId>urn:uuid:67724f3b-38aa-4508-af84-10df715c5b41</correlationId>  
 <requesterNSA>urn:ogf:network:nsa:netherlight</requesterNSA>  
 <providerNSA>urn:ogf:network:nsa:czechlight</providerNSA>  
 </ns7:nsiHeader>  
 </S:Header>  
 <S:Body>  
 <ns5:reserveConfirmed xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/types"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/framework/headers">  
 <reservation>  
 <globalReservationId>urn:ogf:network:service:9cd41aa0-b019-435f-b3dc-075098816737</globalReservationId>  
 <description>This is a test schedule connecting ps-80 to ams-80</description>  
 <connectionId>urn:uuid:deb6aa8d-3e59-4821-9f7f-5c808be42bde</connectionId>  
 <serviceParameters>  
 <schedule>  
 <startTime>2012-05-02T15:47:33.937Z</startTime>  
 <endTime>2012-05-02T15:49:33.945Z</endTime>  
 </schedule>  
 <bandwidth>500</bandwidth>  
 </serviceParameters>  
 <path>  
 <directionality>Bidirectional</directionality>  
 <sourceSTP>  
 <stpId>urn:ogf:network:stp:czechlight.ets:ps-80</stpId>  
 </sourceSTP>  
 <destSTP>  
 <stpId>urn:ogf:network:stp:czechlight.ets:ams-80</stpId>  
 </destSTP>  
 </path>  
 </reservation>  
 </ns5:reserveConfirmed>  
 </S:Body>  
</S:Envelope>

The following message is the HTTP response containing the acknowledgement to the previous ReserveConfirmed primitive.

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Content-Type: text/xml;charset=utf-8

Transfer-Encoding: chunked

Date: Wed, 02 May 2012 15:45:34 GMT

<?xml version='1.0' encoding='UTF-8'?>  
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
 <S:Header>  
 <ns6:nsiHeader xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/framework/headers"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/connection/types">  
 <protocolVersion>http://schemas.ogf.org/nsi/2012/03/connection</protocolVersion>  
 <correlationId>urn:uuid:67724f3b-38aa-4508-af84-10df715c5b41</correlationId>  
 <requesterNSA>urn:ogf:network:nsa:netherlight</requesterNSA>  
 <providerNSA>urn:ogf:network:nsa:czechlight</providerNSA>  
 </ns6:nsiHeader>  
 </S:Header>  
 <S:Body>  
 <ns5:acknowledgment xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"  
 xmlns:ns3="http://www.w3.org/2001/04/xmlenc#"  
 xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"  
 xmlns:ns5="http://schemas.ogf.org/nsi/2012/03/connection/interface"  
 xmlns:ns6="http://schemas.ogf.org/nsi/2012/03/framework/headers"  
 xmlns:ns7="http://schemas.ogf.org/nsi/2012/03/connection/types" />  
 </S:Body>  
</S:Envelope>

# Contributors

Chin Guok, ESNET

Jeroen van der Ham, University of Amsterdam

Radek Krzywania, PSNC

Tomohiro Kudoh, AIST

John MacAuley, Surfnet

Takahiro Miyamoto, KDDI R&D Laboratories

Inder Monga, ESnet

Guy Roberts, DANTE

Jerry Sobieski, NORDUNET

# Glossary

Connection

A Connection is a conduit that transparently moves user information between STPs across a Network. A Connection has a set of properties (for instance, capacity, or authorization, or start time).  These properties, and their allowed range of values, are defined by a Service Definition. A Connection instance on the Transport Plane is identified by a Connection Identifier exchanged on the Service Plane. Connections are unidirectional.

Connection Service

A Connection Service is a service that allows a Requester NSA to request and manage a Connection from a Provider NSA

Connection Service Protocol

A 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.

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.

Edge Point

A network resource that resides at the boundary of an intra-network topology, this may include for example a connector on a distribution frame, a port on an Ethernet switch, or a connector at the end of a fibre.

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 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 Requester NSAs and Provider NSAs.  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 a NSI message based platform capable of supporting a range of Network Services.

NSI Message

A NSI Message is a structured unit of data sent between a Requester NSA and a Provider NSA.

Provision/Provisioning/Provisioned

Provisioning is the process of creating the physical instance of a Connection in the data plane. A Provisioned Connection is ready to carry user data. This term has a formal definition in the CS state-machine.

Requester/ Provider NSA (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 NSA. When an NSA realizes a service, it is called a Provider NSA.  A particular NSA may act in different roles at different interfaces.

Release/Releasing/Released

Releasing is the process of de-provisioning resources on the data-plane. When a Connection is Released on the data-plane, the Reservation is retained. This term has a formal definition in the CS state-machine.

Reserve/Reserving/Reservation

A Provider Agent holds a Connection Reservation. This Reservation has start and end times, ingress and egress STPs and an associated set of additional local resources necessary to build the Connection locally. This term has a formal definition in the CS state-machine.

Service Definition

The Service Definition consists of a set of attributes that formally and explicitly define the complete scope of a service offering. Each provider defines its service with an SD, each request identifies requirements in terms of SD attributes, and each Connection has an associated Service Definition instance.

Service Demarcation Point (SDP)

Service Demarcation Points (STPs) identify a grouping of two Edge Points at the boundary between two Networks.

Service Termination Point (STP)

Service Termination Points (STPs) 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.

Terminate/Terminating/Terminated

Terminating is the process which will completely remove a Reservation and Release any associated Connections. This term has a formal definition in the CS state-machine.

Transport Plane

The Transport Plane contains is the set of transport equipment and associated resources that carry user data through the network.

# Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

# Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

# Full Copyright Notice

Copyright (C) Open Grid Forum (2008-2012). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

# References

1. OGF GFD.173: “Network Service Framework v1.0”, http://www.gridforum.org/documents/GFD.173.pdf
2. IETF RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specification
3. IETF RFC 4122, A Universally Unique IDdentifier (UUID) URN Namespace
4. ITU-T Rec. X.667 Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components
5. ISO/IEC 9834-8:2005 Information technology -- Open Systems Interconnection -- Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components
6. IETF RFC 4655, "A Path Computation Element (PCE)-Based Architecture", http://www.rfc-editor.org/rfc/rfc4655.txt
7. ISO 8601:2000 “Data elements and interchange formats — Information interchange — Representation of dates and times” or xsd dateTime
8. IETF RFC 5905, “Network Time Protocol Version 4: Protocol and Algorithms Specification”, http://tools.ietf.org/html/rfc5905
9. IETF RFC 6453, “A URN Namespace for the Open Grid Forum (OGF)”, <http://tools.ietf.org/html/rfc6453>
10. OGF GFD-CP.191 "Procedure for Registration of Subnamespace Identifiers in the URN:OGF Hierarchy”, http://www.ogf.org/gf/docs/
11. W3C XML “Schema Definition Language (XSD) 1.1 Part 2: Datatypes”, http://www.w3.org/TR/xmlschema11-2/#anyURI