**Network Service Interface Architecture**

Status of This Document

This document provides information to the Grid community on the service interface between a requesting software agent and the provider software agent that offers and delivers a network service. It is intended to describe the processes and environment in which software agents interact to deliver the service(s). Representing applications or other networks, these agents may request certain services of other network agents. Distribution is unlimited.

Copyright Notice

Copyright © Open Grid Forum (2008-2010). All Rights Reserved.

Trademark

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

Abstract

The Network Service Interface 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 Network Service Interface Architecture describes a service plane network topology model, and associated processes and concepts that occur among the Network Service Agents in order to satisfy service requests.

Contents

[1. Context and Overview 2](#_Toc257738118)

[1.1 The Network Service Interface 3](#_Toc257738119)

[1.2 The Network Service Agent 3](#_Toc257738120)

[1.3 The NSI Service Plane 4](#_Toc257738121)

[1.4 Service Abstractions 5](#_Toc257738122)

[2. The NSI Protocol 6](#_Toc257738123)

[2.1 Temporal aspects of services 7](#_Toc257738124)

[3. Representing network resources 8](#_Toc257738125)

[3.1 Resource Abstraction 8](#_Toc257738126)

[3.2 The NSI Topology Model 9](#_Toc257738127)

[3.3 Service Termination Points 10](#_Toc257738128)

[4. The NSI Services 11](#_Toc257738129)

[4.1 NSI Connection Service 11](#_Toc257738130)

[4.1.1 Connection service concepts 11](#_Toc257738131)

[4.1.2 The Service Definition 13](#_Toc257738132)

[4.1.3 The Connection Service States 14](#_Toc257738133)

[4.1.4 Tree and Chain Connection Service Request Processing 14](#_Toc257738134)

[4.1.5 The Path Object 16](#_Toc257738135)

[4.1.6 The Connection Path Algebra 16](#_Toc257738136)

[4.2 Future Services 16](#_Toc257738137)

[5. Appendix A. 17](#_Toc257738138)

[6. Contributors 19](#_Toc257738139)

[7. Glossary 19](#_Toc257738140)

[8. Intellectual Property Statement 20](#_Toc257738141)

[9. Disclaimer 20](#_Toc257738142)

[10. Full Copyright Notice 20](#_Toc257738143)

[11. References 21](#_Toc257738144)

# Context and Overview

Over the last decade, global networks have begun delivering high performance transport services directly to applications that require performance levels or capabilities unavailable in conventional best-effort IP networks. The ability to create connections between a fixed set of ports worldwide, with specific, predictable, and often demanding performance characteristics, enables emerging global collaborations to establish well defined and highly customized network environments to support the end users and their applications. This is particularly true within the Research and Higher Education space and the growing Grid community.

Connections across these transport networks have been historically reserved and provisioned in a variety of ways. The most common approach is manual provisioning by someone (often a network engineer) who is directed by traffic capacity planners. More recently, some networking communities have developed tools and protocols to automate the process of network resource allocation and to allow the user or application to participate directly in the path planning process. These new approaches to automating transport connection provisioning are the basis for the standardization effort being described in this recommendation.

Connection-oriented transport capabilities are being deployed by Research & Education (R&E) providers as well as by commercial providers, and may eventually be implemented in home/ retail networks as deployment progresses.

These automated provisioning systems, while created by different groups, have all developed software based control and/or management agents to regulate access to these resources, to schedule and reserve resources, to trigger or control timely provisioning of the network resources, and to monitor and release resources. These controllers are deployed in two different contexts. One context is application (or Grid) centric, where a network provides a resource to an application. The other context is network centric, where network resources are collaboratively shared among networks to expand or improve network performance or reach. In the former context, a user or application agent is requesting the service of a network provider. In the latter context, one network is interacting with other network(s) to manage these resources – and hopefully deliver a comprehensive and well integrated service portfolio to the user community. This recommendation defines an interface which works in either of these contexts.

The NSI defines several key architectural elements: the Network Service Interface (NSI), the Network Service Agent (NSA), the NSI Protocol, the supporting topology model, and a set of basic services. These concepts are assigned to a nominal Service Plane. The Network Service Interface (NSI) Architecture describes an environment within which network capabilities and components are abstracted and manipulated as “network resources.” Within the NSI Architecture, network resources can be selected, allocated, interrogated, and manipulated by software agents on behalf of networked applications. Network resources and capabilities are presented to the consumer through a set of “network services.” The Network Service Interface presents a simple “one-stop shopping” model for interacting with these services.

The NSI is intended to be an interface that provides an open platform to support a range of network related services and functional capabilities such as topology sharing, path finding, resource reservation, hardware provisioning, and other ancillary services and functions. The initial service defined within this framework is the NSI Connection Service.

This NSI Architecture document describes the broad concepts, models, and key objects necessary to realize the delivery of these services. The NSI Protocol definition describes the specific and detailed messages, associated parameters, transactions, and state transitions that occur to request and deliver network services.

## The Network Service Interface

The “Network Service Interface” (NSI) is the boundary between two communicating software agents: a “Requester Agent” and a “Provider Agent”. These agents interact to realize the delivery of some information or function intrinsic to the network infrastructure. This is a simple concept where the requester agent requests some service, and the provider agent attempts to deliver it (see Figure 1).

Figure 1: NSI interface

While the obvious scenario for the NSI is between a user or application and the network service agent, this requester/provider relationship is also present as service requests are decomposed into subparts and multiple involved networks interact with one another to negotiate global Network Services. The NSI is designed to work in both the inter-operator and application-to- operator scenarios.

## The Network Service Agent

The NSI architecture is based around a software agent called the Network Service Agent (NSA). The NSA provides the interface to manage services, resources and state for a transport network. A network is the group of transport resources which are available to the NSA to deliver services.

In the NSI Architecture, the requester agent and the provider agent are roles assumed by a Network Service Agent (NSA). The software based Network Service Agent may at times act as a requester over one interface, and as a provider at a different interface. Indeed, the NSI services are generally designed as distributed but cooperating services that exist across many networks but which cooperate to fulfill a user’s service request.

The Requestor NSA establishes a trusted session with a Provider NSA. This session forms a channel that may be used to exchange Network Service messages.

The NSA incorporates a number of functional components – some of which may be separately defined NSI service others may be simple NSA internal functions (Figure 2). An example of the former might be a Connection Service or a Topology Service. The NSA is central to the NSI Architecture since all NSI processes are invested in the Network Service Agent.

There is no stated limit to how many trust relationships may exist for a particular NSA, but an NSA is specifically required to support multiple simultaneous sessions. An NSA may spawn multiple processes as long as a coherent single NSA model is preserved. A more detailed treatment of session and process management can be found in the NSI Protocol document.

Figure 2: NSA

The NSI Interface must provide a common framework in which network services can be delivered. To achieve this aim, the NSI Architecture is extensible; it anticipates and expects many possible network services to emerge over time. Examples of anticipated services include a Connection Service to create and manage network connections, a Topology Service to distribute topology information, and/or a Directory Service for registering and mapping names to service layer or transport layer constructs. The Network Service Agents must support these services and functions in order to provide the integrated service envisioned.

## The NSI Service Plane

The relationship between and among NSAs is a concept that is fundamental to the NSI Architecture. These relationships and the constructs and semantics they support define the NSI Service Plane. The NSI Service Plane sessions are a secure and reliable mechanism for transactions and service related communication between NSAs. The Service Plane operates on an abstracted view of the transport plane. In this abstracted view, transport resources within a particular network are treated as a single opaque object under the control of an NSA. In the NSI, many of the basic notions that have traditionally been only transport layer concerns have been elevated and abstracted into generic service layer concepts.

This architecture identifies the NSI as existing on the Service Plane, as shown in Figure 3. In general, the Service Plane relies on the capabilities of the Control Plane and Management Plane to effect changes in the Data Plane, where these planes follow the conventional definitions.

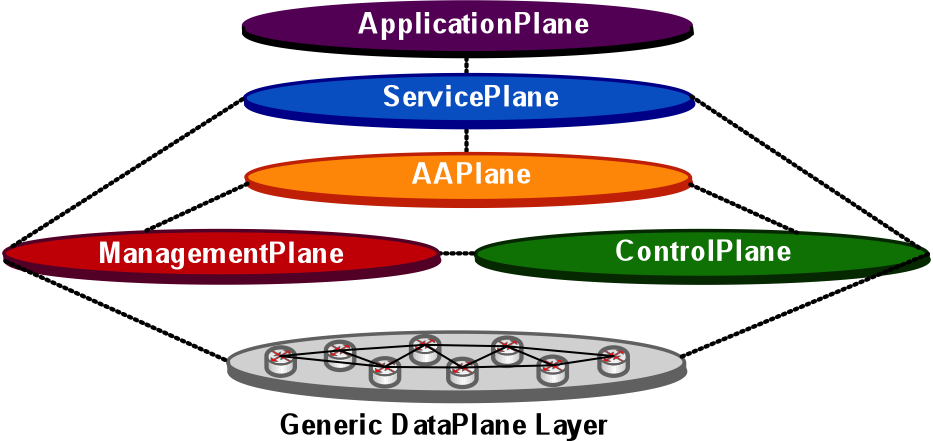


Figure : Location of Service Plane in relation to other planes

Considering the example of the NSI Connection Service, as will be discussed, this has defined a simple, generic - but well bounded -notion of a “connection”, with a set of simple, well defined manipulation rules that operate within a network environment defined by the abstract NSI topology model. These abstractions are exchanged and processed in the NSI Service Plane. Automated semantic processes translate the NSI Service Plane state and events into actual hardware specific configuration changes. The NSI Service Plane contains information about owners, permissions, performance parameters, schedule, and Path information. The Transport Plane is where the Connection is “instantiated” and is where the actual user data payload is carried. The Connection is an NSI Service Plane construct that binds high level service planning results (such as path selection, authorization, reservation, and scheduling) to the low level configuration and management information necessary to instantiate that path in the transport plane.

Figure 4: Provider agent delivers network service

The Network Services are delivered by the Provider NSA via the NRMs managment of its local network resources.

## Service Abstractions

In this section, the separation of the conceptual presentation of a network service from the physical implementation of the service is described.

First, the NSI Service Layer presents an abstracted model of transport services. This abstraction reduces or hides many of the real-world complexities of delivering a particular transport service. For instance, the NSI Connection Service takes a rather complex data transport circuit and presents it as if it were a simple pipe of a certain size between two locations. The user only need specify the ends, and the diameter of the pipe. The user places data in one end of the pipe and it emerges some time later from the other end in the same manner. This abstract concept is a simplified and convenient means of presenting the key functional aspects of the service object while hiding most or all of technical details that are in general not relevant to the application. In practice, the abstraction of a service presents a basic set of service primitives and a bounded set of parameters on those primitives that fully define what will be delivered as result of the service. To make this point, the NSI Connection Service provides primitives such as reserve() and cancel() that perform very clear operations on a “connection”, and these primitives are carefully bounded by the parameters associated with each – such as capacity, or end points, or start time. The details of coordinating schedules and circuit provisioning information across possibly many networks is hidden.

In the NSI Connection Service there are two mechanisms used to formalize the abstractions of a “connection”. First is a “Service Definition”. The Service Definition formally describes each aspect of a service. Each aspect is denoted by a “Parameter” and each Parameter is assigned a specific and bounded set, or range, of values. For instance, an “Ethernet Transport Service” might define a Parameter called “Capacity” that defines a range of allowable service capacities between 1 Mbps and 10 Gbps. This is a very important formalization of the Service offering and so is included here as part of the NSI Architecture. The Service Definition has its roots and most immediate application in definition of the NSI Connection Service offering(s), and for NSI v1.0 that is the sole purpose for which the Service Definition is adopted. (Note that the notion of formal service specifications is still a widely researched topic with new application to emerging network services from Connections to Topology to Monitoring. Further exploration and refinement of this helpful concept within the NSI Architecture will be a continuing effort in NSI futures.)

The second item that gets abstract in this document is the service instance itself. The “Service” defines the full set of capabilities that are offered to requesters. The “Service Instance” defines one specific instance of the service. Put another way, the Service Instance is the object or capability that is delivered to the user in fulfillment of a service request. As an example, a Service Instance for a Connection Service is a particular connection. A service instance for a topology request might be a snapshot of the local topology database. The nature of a Service and a Service Instance is specific to the function the service is intended to perform.

In the NSI Architecture, these Services exist in the form of service specific agents known to, or incorporated within, the NSA. The NSI Protocol provides an extensible framework for the definition and incorporation of network services. The NSI has defined only one initial service: the Connection Service to meet the emerging need for automated creation and management of network connections.

# The NSI Protocol

Network Services are delivered through the use of the NSI Protocol. This protocol defines the constructs, state machines, messages, and parameters associated with the NSI services model. An NSA, by definition, is an agent that implements the NSI Protocol.

The NSI Protocol requires a “trust relationship” between NSAs. These trust relationships mean that each NSA believes the other to be whom they claim to be (authentication) and that both NSAs are willing to accept service requests from the other and act to satisfy them (authorization). Further, there must be a comfort level that the messages have not been tampered with, and optionally that they have not been exposed to unauthorized/untrusted third parties.

These trust relationships can exist in one of two modes: First, for high volume and/or persistent peering relationships, an authenticated, authorized, secure (encrypted) and reliable session can be established between the NSAs. Traffic passed across such a session is known to be trusted and can proceed directly to the service handler. The second mode is to employ a more message based trust framework such as Web Services. This message based form is more appropriate for occasional messaging as might occur between an application agent and various provider NSAs.

The base NSI Protocol handler recognizes “service types”, and “messages” between NSAs. The Protocol examines each message received for a “service identifier” and forwards that message to the appropriate service specific agent.

Each NSI service defines a “service instance” and a set of “service primitives” that operate within the context of a service instance. This service instance is an independent, uniquely identifiable deliverable unit of the service. The NSI Connection Service refers to a particular connection as a “service instance”. A topology distribution service may define an instance to be a particular topology graph, or a topology transaction such as a full dump or incremental update. In general, a service specific state machine allocated and associated with each service instance, and the service primitives drive the transitions of that state machine. A service primitive may require a sequence of messages or even its own state machine to affect an exchange of messages

Service Instances are processed asynchronously with respect to other service instances. For example, one connection may transition from reserving, to scheduled, to in-service, to release at a vastly different speed than another connection established by the same service agent.

The service specific state machine is defined by the service. For example, the state machine associated with establishing a connection is dramatically different than the state machine associated with distributing topology. NSI Protocol messages are the smallest protocol data unit. Each message envelope contains sufficient information to route the message to the proper network service agent, followed by sufficient service specific information to associate the request to an appropriate service instance and to identify the service primitive.

The NSI Protocol Specification provides a detailed description of the NSI protocol.

## Temporal aspects of services

Services in which resources are dynamically requested, reserved and provisioned, temporal aspects need to be considered. When resources are sought by a requester NSA from a provider NSA, a service instance is created and an identifier is assigned to that service instance. Then, according to the parameters of the request (i.e. its service definition), the provider NSA identifies a set of available resources which satisfy the request and associates them to the instance (reservation). The resources are provisioned and released at some points in the temporal axis. In the context of network services, timing of resource requests, allocation and provisioning can be classified as “on-demand”, “advance reservation” and “immediate reservation”. NSI interface must support advanced reservation, and may support immediate reservation.

**Advance reservation (book ahead)**

The required resources and the provisioning start and end time are all specified in an advanced reservation request. The request is processed by a scheduler, and the scheduler finds a set of resources available for the requested duration and allocates them to the request to create a reservation. If the scheduler cannot find available set of resources which satisfies the request, the request is denied. This scheduling process is part of path finding in the connection service. The resources are reserved during the reservation period, and when the reservation period begins, the resources are instantiated. A reservation database (i.e. calendar) should be maintained by the scheduler or resource managers, and referred and updated by the scheduler. In advanced reservation, scheduling is done when a request is received.

Figure : Advanced reservation.

**Immediate reservation**

A requester NSA may request immediate provisioning (i.e. provisioning as soon as possible) and specify the end time of the provisioning. This is called immediate reservation. Note that this is different from the *pure* “on-demand” in which the end time of a provisioning is not determined, “immediate reservation” does not occupy resources infinitely. In this sense, the immediate reservation is basically same as the advance reservation. However, the start time varies depending on the time required to provision the resources after a request is issued. If resources from multiple networks are involved, it may be not easy to synchronize the start time of provisioning of resources. Such temporal aspects should be carefully considered in the implementation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Timing of  Scheduling  (to request) | Timing of  provisioning (to scheduling) | End time of provisioning | Maintained states (DB) |
| Immediate Reservation | Immediately | Immediately | Determined at the time of scheduling | Current state of resources, Reservation database |
| Advance reservation | Immediately/ when availability of resources changed | Scheduled | Determined at the time of scheduling | Current state of resources, Reservation database |

Table x type of request and provisioning

# Representing network resources

## Resource Abstraction

The network itself is abstracted into a simplified topology model based on STPs and Networks. This model only exposes salient network characteristics at each level of scale. Indeed, many constructs within this abstracted topology are themselves abstracted representations of other objects. Networks are simple abstracted representations of large interoperating telecommunications infrastructures. But knowing the details of each of the network devices in every network around the globe is not necessary to most network service functions. For instance, we may only be interested in knowing which network(s) are along a path to some destination and what agent(s) are responsible for reserving paths through them. Only certain information is necessary to make, say, a path selection decision, and so we can abstract the topology information to reflect those relationships or requirements.

The NSI Protocol is intended to be a light weight, and extensible. It is designed to allow implementations to be self contained and easily deployed and managed. The protocol stack and the intrinsic protocol functions (such as AA interfaces, configurators, etc) may be implemented in a variety of ways (such as a basic C/C++ library). It is also expected to be implement within a Web Services environment to aid in developing web based tools and processes.

The process of creating an abstraction of resources is performed and owned locally by (by who?). The NSAs then operates on this abstracted representation.

## The NSI Topology Model

In order to develop and define network services – services that interact with and manipulate network resources - the NSI Architecture must posit a basic network model.– i.e. a minimal set of objects and rules that describe a simplified generic network. With regard to the NSI, we are most concerned with the inter-domain network topology. (see Figure 5)

Figure 5: Inter-Domain Topology

The NSI transport topology model is an “abstracted” topology. That is, it captures the logical attributes of the network rather than just the nominal physical arrangement of the hardware. These attributes include representations of administrative boundaries and/or logical relationships such as federations of networks. The primary purpose of the NSI topology model is to describe how *Networks* are interconnected. The NSI Architecture ascribes the management of low level Transport Plane components to an NSA function called the Network Resource Manager (NRM). The NSA is responsible for the high level interactions between Networks via the NSI Protocol. The NRM is shown as the green part of the NSA in Figure 6.

Figure 6: NSI interfaces allow inter-domain service negotiation

The formal representation of topology is out of scope for the NSI, but topology none-the-less plays an important role defining how the NSI service plane interacts to establish global service reach. Therefore, we discuss topology here in order to define key concepts that the NSI relies upon to function.

NSI supports the administrative grouping of transport layer resources into a single topology object called a *Network*. Networks interconnect with other Networks via edge constructs called *Ports*, as in the NML Node. However, Network Ports at the NSI network level are logical constructs and may not have a direct physical analog in the transport plane. NSI Ports differentiate inter-domain transport links originating or terminating in a particular Network. It is the responsibility of the NSI service agent(s) to define a valid mapping function to relate the inter-network relationships to the actual transport devices.

By aggregating detailed transport topology into a single Network, or by grouping several Networks together to form a federated Network object, the global network topology may be reduced substantially. How such a federation is implemented and the resulting simplified inter- Network topology map is out of scope for at least this version of NSI.. Successful implementation for a particular deployment will allow Pathfinders to inexpensively compute coarse grained path(s) between any pair of networks. Each NSA along the candidate path is then consulted to reserve and confirm the resources. (For most of the NSI discussion (certainly for V1.0 inter-domain discussion) we assume each Network has a transfer function that is a simple non-blocking any-port-to-any-port crossbar switching function. Any other path constraints are managed by the Resource Manager.)

The important aspects of this topology discussion is to note that there are two levels of pathfinding recognized by the NSI Architecture: the inter-domain pathfinding which is concerned with choosing a coarse Path across the global topology of Networks, and the intra-domain pathfinding concerned with the transport resources within the Network, Pathfinding algorithms and processes are generally speaking out-of-scope for NSI, However, since processes such as Connection Request processing necessarily involves and depends upon network selection, pathfinding is often referred to in order to obtain a path or information about a path. The upper level NSI coarse grained path selection does not skip the low level path planning and reservation phases, but it effectively prunes the search space early in the process to produce a “good bet” path that has a high likelihood of success.

From a global perspective, hiding detailed transport topology within an opaque Network object substantially reduces the size and complexity of the topology information base. This has positive implications for coherence and convergence, for dynamic topology distribution, pathfinding efficiency, and for scalability in the global environment. It has the less desirable effect of reducing optimality – it becomes increasingly difficult to choose a resource efficient path. The trade off is an issue of pragmatism, and will be steered by best practices as the experience base improves.

## Service Termination Points

The NSI Architecture adopts a generalized notion of a Service Termination Point (STP). An STP names a topological location that acts as the junction between the start (or ingress) of a Connection and the access link that presents the user data to the connection. A similar junction terminates the connection and is likewise identified by another STP. STPs are an important abstraction: In NSI, they are uni-directional constructs, typically associated with Ports in the topology. Whether an STP functions as an ingress point or an egress point is defined by the flow polarity of the associated Port, and which side of the junction is the user side, and which side is the network side. It follows then, that since both sides of the STP junction are exactly the same, if a connection could be terminated on one side, then another connection could be originated on the other. The STP can in fact function as both an ingress point for one connection and an egress point for another connection – simultaneously. Two such connections that share a single STP are said to be concatenated. Two concatenated connections appear as a single end-to-end transport plane data path to the user payload data. This double-duty application of an STP allows an STP to also be specified as an intermediate transit-point of a path or connection, i.e a point through which the connection must pass.

As alluded to above, this definition of a Service Termination Point correlates very nicely with the topological definition of the Port object presented earlier. The Port’s unidirectional nature and bi-connectivity (support for an input connection and an output connection) make it the appropriate topological object to use in mapping connections across the transport plane. And in fact, the most common use of STPs is in path specifications where they do map to Port objects. However, the STP construct and the Port object are not strictly synonymous.

These abstracted properties of Connections, and STPs are discussed in greater detail later in this document in conjunction with the Connection Service specification.

An STP is a symbolic reference, i.e. it is comprised of a parsable alphanumeric string containing two components: 1) a Network identifier string in the higher order portion, and 2) a local STP identifier in the lower order portion. An STP must always resolve to a specific topological Port object as defined in the NSI topology. It is an implementation decision as to how to map that NSI Port identification to the corresponding resource in the physical transport plane.

It is important to note that the NSI inter-network topology model is NOT a standard nor does it imply that an NSI implementation must adopt specifically such a topology database in the code. The NSI topology model is simply an abstraction that allows this specification to describe the architecture, the set of objects, agents, and algorithms that the NSI requires to function fully. The NSI relies on “network” domains (similar to BGP Autonomous Systems) to hide and/or summarize network topology information. The NSI Architecture defines a Network as the set of network resources under management of a particular NSA. This NSA is then “authoritative” for all resources that report up to it – i.e. no other agent (NSI or otherwise) is allowed to manage any of the resources delegate to this NSA, and all requests for NSI resources in this network must be submitted to the local NSA. The local NSA will dissect the request and forward subparts to the underlying Resource Managers.

The NSI specification also allows for federations of Networks. For instance, an arbitrary set of Networks may band together under NSI rules and peer exclusively with a single parent “Federation NSA”. The parent federation NSA may have no transport resources of its own – just those resources under management of the children NSAs. Service requests will flow along the trusted sessions hierarchically among NSAs, even if the transport connection itself takes a different route through the transport infrastructure.

# The NSI Services

The NSI Protocol is designed to mediate trust sessions and deliver messages between many NSA based services. NSI V1.0 stipulates a single NSI Service: The NSI Connection Service. It is described below.

## NSI Connection Service

### Connection service concepts

The NSI Connection Service is the Network service that manages Network Connections.

There is a growing requirement to integrate customized networks resources into existing grid resources pools and applications. The ability to manage network connections effectively and easily by the grid community is perhaps the most pressing driver for the OGF NSI specification effort. This particular requirement has had a significant influence on design decisions and prioritization of capabilities incorporated into the NSI Architecture.

The NSI Connection Service (CS) reserves, schedules, and instantiates Connections. An NSI Connection Service “instance” is a single channel, unidirectional, point-to-point construct.

The use case identified for the NSI Connection Service will be to support a high capacity, highly asymmetric data flow such as occur in large file transfers or real-time streaming of digital media content. The reserving of symmetric high capacity back channels, used sparsely if at all, for acknowledgement traffic for these data driven applications would waste enormous network capacity. Indeed, it is highly unlikely that even a low speed back channel connection would be necessary for such acknowledgements, or that such a channel would need to be allocated along the inverse path of the primary connection. A single uni-directional point-to-point connection is the atomic service object delivered via the NSI Connection Service. Where bi-directional connectivity is required, two separate service requests can be submitted by the requester.

Figure : Anatomy of a Connection

As illustrated in Figure 6, 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. The network components that present the user data to the ingress point or carry the user data away from the egress point are the access sections. The network infrastructure that carries user data from the ingress point through the network to the egress point is the transport section. The end of the transport section or the junction between the transport section and the access section is called the Service Termination Point (STP).

The user data (the “payload data”) is carried 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 switch to switch through the network. It is important to distinguish between a) the access framing protocols, b) the transport framing protocols, and c) the user payload data carried inside each of these protocols. *It is the user payload data stream that must be preserved end to end in a connection.*

The end points of a connection are locations in the NSI topology where the Connection Service can terminate a Connection. Thus, they are referred to as “Service Termination Points.” (STPs) The access framing is defined implicitly or explicitly with the specification of the end points. (While this sounds trite, it should be understood that not every location in the network is able to terminate a particular service. E.g. you cannot terminate an Ethernet connection at a SDH port.)

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. This is another example of how abstraction separates the user perception of a “connection” from that of the provider. While this abstraction simplifies the service concept, the NSI Architecture allows the omniscient requester to participate in these connection planning decisions.

This abstracted concept of a “Connection” is, in essence a data pipe. The pipe extends from an ingress point to an egress point. The pipe has a non-zero length (latency). The user data is preserved end to end. Data flows only one way inside the pipe. The requester can specify some few parameters that describe the form and flow of that user data such as source and destination, capacity, start and end times, etc. These parameters are configured and implemented at the connection end points. The only requirement on the transport section (the network) and the access section (the user) is that these connection constraints, once reserved, confirmed, and instantiated, are not violated by either party.

The service levels offered by a network will be defined by the Service Definition as offered by any given network.

### The Service Definition

The Service Definition is a set of attributes that formally and explicitly define the complete scope of a service offering. In particular, the NSI Connection Service uses the Service Definition as a baseline set of parameters to bound the scope of the service that will be offered to requesters.

The Service Definition specifies the complete set of service parameters that define a service instance. The Service Definition also describes the *range* of allowed values for each service parameter, and a default value can be specified. The parameters in the Service Definition form a kind of template that the service request must fill in. I.e. A service request must fill in the template with an explicit value - or a default value taken from the service definition - for each parameter of the service it is requesting. A service instance is then the service capability that results when all parameters of a service have been fully determined.

If a service request describes a service instance that lies within the bounds of the set of defined service parameters, then it forms a valid request. Further, by merit of the comprehensive service parameters in the Service Definition, and the instance specific parameters present in the request, the bounds of the network commitment are formalized and explicit to both the requester and the provider. This explicitly defined service commitment allows the user to verify the delivered service and determine if it is meeting the commitment. It also acts as objective criteria for determining the status of a connection: “up and available” meaning it is operating within the committed service levels, and “down” meaning the connection is not operating within the committed service level.

The service definition provides a publicly available description of the service, and should be made available in a native language document that the users can reference in developing or configuring their applications. The users should consult this service definition in order to understand what service levels are available to them within a given service offering.

It is important to stress one more aspect of a service definition; if a parameter is not identified within the service definition document, then the user can make no inference about its presence or value in the service. For instance, if a service definition has no jitter specifications, the user can make no predictions or assumptions about the jitter. And the network has made no commitments regarding jitter. A request on Monday might have excellent jitter characteristics, and the exact same request submitted Tuesday might have horrid jitter characteristics. As long as the request constraints on both requests were met, these are – from a formal service perspective – properly performing and identical service instances.

The converse is also true. The network should be very careful about how it defines service parameters. For instance, an Ethernet service may define connection capacities in “bits per second” (bps). On its face, one might construe that a 1 Gbps connection would accept bits at 1 billion bits each second measured over any one second period. However, if this 1 Gbps connection is provisioned over a 10 Gbps network link, this interpretation would allow a 100 millisecond burst at 10 Gbps followed by a 900 millisecond quiescent period. Such a burst of 125 megaBytes can easily induce buffer overruns and packet discards on interfaces along the connection path. And yet the user would have been perfectly within their performance profile. The implication here is that simple fixed capacity connections in asynchronous packet transport networks requires sophisticated and detailed planning in order to guarantee service capabilities.

The Connection Service Definition and the Connection Service Request are tightly integrated.

### The Connection Service States

The states of a connection relate to the life cycle of the connection. In the NSI, a connection goes through five phases: Scheduling, Reserved, Provisioning, In-Service, Released.

First, a request is submitted to a provider thus beginning the life cycle of the connection within that provider Agent’s network. This first phase is called the “Scheduling” phase. It includes path selection and resource reservation. In the NSI, path selection includes future scheduling as well as the performance and authorization checking. Once the scheduling phase is complete, the connection goes into a “Reserved” state.

When the service start time arrives, the connection goes into a “Provisioning” phase. Provisioning is where the connection is physically instantiated by configuring each device along the path to reflect the path plan developed and reserved in the Reservation phase.

Once provisioning is complete, the connection then moves into an “In-Service” state and the user is notified that the connection is ready for use. The In-Service phase is where user data is allowed to transit the connection.

When the connection is no longer needed (or the scheduled time expires) the connection is “Released”. The Release phase is where each network along the path is informed of the Release event and resources associated with the connection are released back to the available pool. Upon entering the release phase, the connection will no longer pass traffic. When the Release has completed, the connection object is deleted from the service layer.

### Tree and Chain Connection Service Request Processing

The NSA builds connections across networks. Connections extending across multiple networks can be constructed by concatenating connections built across the individual networks. The one prerequisite for building concatenated connections is choosing appropriate STPs such that the egress STP of one connection corresponds directly with the ingress STP of the successive connection. These STPs are the inter-domain transit points between the two inter-connected networks. Note that here the term inter-network is synonymous with inter-domain.

The choice of which sequence of networks to use is a path finding function and dependent upon topology and information being available to the local pathfinder to choose a candidate inter-domain path. While the end-to-end concatenated path is not confirmed until all individual constituent connections have been reserved and confirmed, once a set of inter-domain transit points is chosen, the connection requests corresponding to each segment is can be issued simultaneously and directly to the corresponding networks. This is called a “Tree” model reservation process.

Alternatively, if the local NSA does not have sufficient topology information or authorization credentials to identify and interact directly with all the downstream networks, the local NSA can simply choose a neighbor network as the next hop, and using the interconnect STP as the ingress point, forward a request to that next hop NSA for handling. This conventional hop-by-hop approach is called the “Chain” model approach.

The reservation process, in general, involves a constraint-based search of the topology for a set of contiguous resources meeting the constraints specified in the user Connection Request. As resources conforming to these constraints are identified, they are reserved in an atomic compare-reserve process. While these are, strictly speaking, part of Path Finding and outside the scope of NSI, it was considered important that the Interface be able to support both styles of reservation – the former being more traditionally found in existing protocols and intra-domain topologies, and the latter providing more control to the requester regarding path selection.

Chain style processing reserves resources sequentially beginning at the source STP and working hop by hop successively through each downstream network to the destination. The path computation requires only a simple next hop reachability calculation (though more sophisticated path finders can be implemented), and no resources are reserved until the prefix path has been confirmed. It is highly distributed, scales well and is robust. But it does enforce a provider centric model that hides or delegates much (if not all) network provisioning decisions.

The Tree model processing computes a course grained inter-domain path first. It uses that network path vector to decompose the connection request into several concatenated connection segments. This decomposition process, while requiring more topology information and a more sophisticated pathfinder, enables the NSA to reserve the segments in parallel via direct interaction with the respective networks. The tree model exposes many new capabilities directly to the user at the cost of significant increase in protocol and operational complexity.

In both the tree style processing and the chain style processing, the end-to-end connection cannot be confirmed until all of the constituent connection segments have been successfully reserved and confirmed. Which model will be more effective is unclear at this time and will likely be directly related to complexity of topology distribution and path analysis, robustness, authorization schemes, request volume, network diameter, utilization density, cost, ease of use, and reach (to name just a few actors.)

Both the Tree and Chain model reduce to a constraint-based search over a topology to build a k-preferred path tree. Both can accept requester guidance in path selection through the inclusion of intermediate transit points in the connection request (discussed further under Path Objects). The method, tree or chain, used to process a request is made exclusively in the requester NSA. The requesting agent implements a Tree model process by submitting individual requests for each connection segment. These individual segment requests are processed asynchronously and in parallel. The requesting agent implements a Chain model by allocating a path through the local network, and then forwarding the request to a neighbor domain to resolve the remaining downstream portion of the connection.

### The Path Object

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

A Path Object associated with a confirmed Connection contains, or references, a significant amount of information regarding the user, the source or destination of flows, the global topology, and internal detail of specific networks, etc. This Path information may pose a security or privacy issue to the user or the involved networks, or may just be considered proprietary information. Within the NSI, access to such information is considered a policy decision of each agent involved. Therefore, Path information is available to external agents via an authorized Query() primitive to the Connection Service.

The provider NSA is responsible for maintaining, among other things, a Path describing the fully specified path for any Connection reserved across its network. In order to protect the PO, the provider NSA must store the Path locally and return a redacted Path containing a list of STPs, and/or Named Path, specified in order, according to its internal authorization policies.

Since Connection Requests submitted to other NSAs may return a Path identifier rather than a Path Object itself, there must be means for distinguishing the two and a clear understanding of how a path object fits into the path algebra. Since the Connection Request segmentation processing is tree-like, it follows that the reserved Path Objects will also be tree-like. So a Path Object must be able to contain not just directly referenced STP Names, but must be able to contained Named POs as well. A Path Object then consists of a list of objects that either directly or indirectly resolve to topological points. For named POs, the NSA that owns the Named PO must also maintain authorization association(s) for the PO. (This pointer to a Path is a new requirement –needs to be agreed)

### The Connection Path Algebra

An algebra is a set of rules for symbolically manipulating objects. The NSI Architecture defines a *path algebra* to symbolically describe the operations performed on paths and connections. The path algebra can be used to describe how connection requests can be iteratively decomposed into component segments as the request is processed down the service tree. The path algebra insures that by choosing appropriate termination points for the component segments, that the resulting set of segments will form a single continuous data path. (In this context, it is useful to remind the reader that the NSI definition of a Connection is a single channel, unidirectional, point to point data path. A “path” then is a contiguous sequence of points and/or edges that are visited on a tour of the graph.)

Within this document, the path algebra is referenced due to its utility in discussing and analyzing the manner in which the NSI Connection Service handles connection requests. The NSI path algebra is described in more detail in Appendix A: “A Path Algebra for Describing NSI Path Operations”.

## Future Services

The NSI defines a framework that will allow future Network Services to be added. The framework is defined in such a way that allows each Network Service to be independent.

# Appendix A.

This belongs in the protocol doc:

The NSI defines a service calls corresponding to the Reservation primitive, the Provision primitive, and the Release primitive. Other service calls are defined for Query and Notify which interrogate the status of a connection, and register a callback routine to catch certain events related to the connection. These primitives are discussed briefly below, and in more detail in the NSI Protocol Specification.

**Reserve()** - The Reserve() function is where the connection life cycle begins. A Requester NSA develops a Connection Request based upon application requirements and conforming to an available and appropriate Service Definition. The requester NSA (the Requester Agent – “RA”) submits the connection request to a provider NSA (the Provider Agent – “PA”). The PA must do some basic qualification of the request to insure that it conforms to the selected Service definition. Upon passing SD conformance, the request is passed to a “PathFinder”. The PathFinder will decompose, or “segment”, the request into segments that local NSA must reserve, and segments that must be allocated by other remote network NSAs. This begins with the ingress STP and egress STP and takes into account any intermediate transit STPs that may be specified in the request. The sub-requests are recursively issued to the remote NSAs to be reserved. (The process by which the PathFinder segments the request and issues sub-requests is not strictly part of the Network Service Interface definition but this document does acknowledge the “Tree” and “Chain” dichotomy in path planning and reservations. A brief diversion into Path Planning, Tree/Chain request handling, and Connection Manipulation Algebra is described below. ) As each sub-request is processed it is also reserved. So at each level of the service request tree, a reservation for that sub-portion is confirmed. The NSI architecture supports book-ahead scheduling as one set of constraints (Start Time/Duration) that a connection request may specify. Upon a Reserve\_Complete() confirmation sent back to the requesting agent, that connection is said to be “scheduled”.

**Provision()** - The Provision() function is invoked when the scheduled start time arrives. The function can be called by the provider agent or the requester agent and is sent up and down the request processing tree to all NSAs associated with the request. The Provision() function initiates the reconfiguration of the hardware along a connection path to reflect the path reservation. This is where the connection is actually built, or “instantiated”. Upon successfully provisioning all the sub-requests associated with a connection, the local NSA sends a Provision\_Complete() message up the request tree. Upon sending (or receiving) a Provision\_Complete(), the connection is transitioned to “In Service” within the sending (or receiving) NSA. It is conceivable that a Provision() function may be initiated at several locations within the request tree at once. The first received Provision() request will be acted upon, and the other silently discarded.

**Release()** – The release function is responsible for deallocation of all resources associated with a connection request. A normal Release begins at the originating user NSA and proceeds down the request processing tree. However, certain events can cause a Release to be initiated by authorized NSAs deeper within the request tree. A Release() function can be invoked by either the RA or the PA and should be sent to all NSAs associated with the connection. Upon releasing the resources, an NSA is required to send a Release\_Complete() up the request tree. The connection object remains in existence until the Release\_Complete() is issued up the tree over to the appropriate trusted NSA. (I.e. the connection object remains in existance until the parent NSA has been notified of the complete release.)

**Query()** – The Query() primitive is used to interrogate status of a connection. As with all Connection Request primitives, the request is authorized relative to the connection being queried and the information being requested. For instance a Query() might ask for the current state of the connection, which would be legitimate for the owner to ask, but perhaps not another user. Some information may not be accessible even to the owner – such as path details – without a higher or different authorization policy.

**Notify()** – This primitive registers a process to receive notification when certain events associated with a connection occur. For instance, an application may wish to be notified when the start time for a connection is reached, or to be notified if a connection is interrupted for any reason.

In the NSI, a connection reservation (broadly construed to mean entire life cycle of the connection) holds a very high place in terms of service commitment. A confirmed reservation (connection) should be considered a hard commitment by the provider agent to the requesting agent and should never be interfered with except in the most desperate situations. Doing so has affects upon many networks and globally integrated applications. In the NSI, there is no notion of pre-emption or priority that can undo a confirmed reservation. Further, it is required that when a reservation is confirmed, that the connection path has been fully specified and all necessary resources allocated. It is the responsibility of the network NSA to maintain a persistent information base adequate to insure that the life cycle of any confirmed reservation is honored.

In the NSI, a confirmed reservation can only be canceled by the original authorized requester. However, there may be times when the network operations and/or engineering staff may need to override a reservation in order to maintain the health or integrity of the network, or an unforeseen resource failure causes a reservation to be inoperative. In this scenario, the network administrator must be able to, at a minimum, cancel a reservation in its entirety. Functionally speaking, a “cancel” is the same as a “release”. Releasing any part of a reservation will invalidate the entire end-to-end reservation – thus the end-to-end resources should be released to make them available to other requests and to insure they are not unfairly charged to the hapless user who’s reservation just got burned. The network administrators along the path of a connection only have override authority within their own networks, and may have little or no knowledge of the end-to-end path for the reservation. Tree and Chain model reservations require different NSA notification paths to release upstream and downstream network resources. NSI v1.0 requires that an NSA shall send a Release primitive to all other NSAs associated with the local reservation when a cancelation has occurred. This will initiate a normal Release process (with a cancelation cause). Since only the user and a local administrator are allowed to cancel a reservation, this cancelation requirement requires that each NSA recognize a “network administrator” user (an authorization policy) in the peer domain across the trusted session. This remote network administrator is the only other authorization policy besides the reservation owner and the local NSA itself that has authority to Release a reservation in a local domain.

Resource reservation is atomic, so there is never a conflict – a requested resource is either reserved or it is not. However, resources are not infinite. It is possible that a reservation may fail at some point due to resource exhaustion. The process of backing up to try alternate paths around a blocking condition is called “crank back”. Crank back can be a complex process in multi-layer, multi-domain networks. This document does not make any recommendations or requirements on how crank back is performed, except to say that alternate paths should be explored – within the constraints of the connection request - before returning a failure status to the requesting agent.

Connection modification is not specified in this version of the NSI definition.

The reserve function, at a minimum, must specify 1) the Service Identifier, 2) ingress point and the egress point for the connection, and 3) Authorization Credentials. All other possible parameters are taken from the Service Definition referenced by the Service Identifier. The request may override any of these defaulted parameters by including them in the request.

# Contributors

Jerry Sobieski, NORDUNET

John Vollbrecht, Internet2

Guy Roberts, DANTE

Inder Monga, ESnet

Tomohiro Kudoh, AIST

# Glossary

Connection

A *Connection*is a conduit that transparently moves user information across a *Network*from an ingress point to an egress point. 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 is a particular *Connection*, identified by a *Connection Identifier*.

Connection Identifier

A *Connection Identifier* is a label unique to an NSI interface which can be used to identify a *Connection*for the purposes of request, instantiation and management.

Connection Service

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

Control and Management Planes

The *Control Plane* and/or *Management Plane* are not defined in this document, but follow common usage.

Network  
A*Network*includes all of the transport resources that are managed by a single NSA.

Network Resource Manager (NRM)

The *Network Resource Manager* owns a particular set of transport resources and has ultimate responsibility for authorizing and managing the use of these resources.

Network Service

A *Network Service* is an abstract notion that must be implemented by a concrete network service agent (NSA). The *Network Service*is the service characterized by the set of functionality that is provided in an NSA.

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*.

NSI Message

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

Path

A *Path*is an ordered list of *Routing Objects.*

Requester/ Provider NSA

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*.

Routing Object

A *Routing Object* may include the following transport resources

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 set *Network Service Agents* communicating using *Network Service Interfaces*.

Topology Rename

The *Topology*resides in the Service Plane.  The *Topology*describes both the physical resources and their interconnection as well as the non-physical groupings of various components

Transport Plane

The *Transport Plane* contains is the set of physical resources that transport user data through the network.  The *Transport Plane* forms the substrate over which Connections are allocated and provisioned.

# 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 (2008-2010). 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