GWD-R, GWD-I or GWD-C
NSI-WG
nsi-wg@ogf.org

John MacAuley, ESnet

February 18, 2014

Network Service Interface Discovery Service

Status of This Document

Grid Working Document (GWD)

Copyright Notice

Abstract

This document describes the Network Service Interface (NSI) Discovery Service version 1.0, a simple protocol supporting the distribution of meta-data documents throughout an interconnected network of Network Service Agents (NSA) operating within the Network Services Framework (NSF). This protocol addresses the key issue of dynamic data distribution within an NSI network by providing a flooding based protocol for exchange of documents published by NSA within the network. By abstracting the protocol used for exchange of the data from the data itself, a more generic protocol is provided which meets the requirements for distribution of NSA Discovery documents, NSI Topology documents, and NSI Service Definition documents.

Notational Conventions

The key words ‘MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC 2119 [BRADNER], except that the words do not appear in uppercase.

Contents

1 Introduction

Within the Network Services Framework (NSF) [OGF NSF] the Network Services Agent (NSA) is an entity that offers network services. Peer NSA entities communicate using the Network Service Interface (NSI) protocols, a suite of individual protocols providing the infrastructure needed to offer network services. Part of these network services is supporting data documents to which a client requires access in order to properly utilize the offered service. One such document is the NSA Discovery Document [OGF NSI-ND], which is a metadata schema designed to enable self-description of all NSI services and associated protocol interfaces offered by these NSA. Other information relating to the NSA itself, such as software version, administrative contacts, location, peering, and managed networks is also defined as part of the meta-data profile.

This type of dynamic data discovery mechanism is an important aspect of any large-scale distributed system. By making the NSI protocol and its agents more self-descriptive, new documents, features, protocols, or protocol versions can be added to agents within the network and then be discovered by peer agents through this meta-data service. As new features come on line, agents supporting the capabilities can discover compatible peer agents, and then negotiate use of these new features, while older versions of agents within the network remain unaffected. Similarly, newer versions of agents can still negotiate features and communicate with older agent versions using mutually supported versions of the protocol as described in the discovered meta-data.

The NSI Discovery Service is part of the NSF suite of protocols, and is a simple peer-to-peer flooding protocol for exchange and distribution of these types of data documents between NSA within the interconnected network or "*document space*". It supports both polling and subscription based notification mechanisms for exchange of documents. For the purpose of this description, a *requester* is any application or Network Service Agent (NSA) that is participating as a client in the discovery protocol (client role). A *provider* is any Network Service Agent (NSA) that is participating in the protocol as a server for the document space (server role). NSA can participate in both the requester and provider roles of the discovery protocol.

A requester utilizes the provider's Discovery Service API to query documents stored within the document space. The requester can also subscribe for document discovery and documents updates within the document space. There are also Discovery Service API to publish, update, and delete documents to/from a local provider.

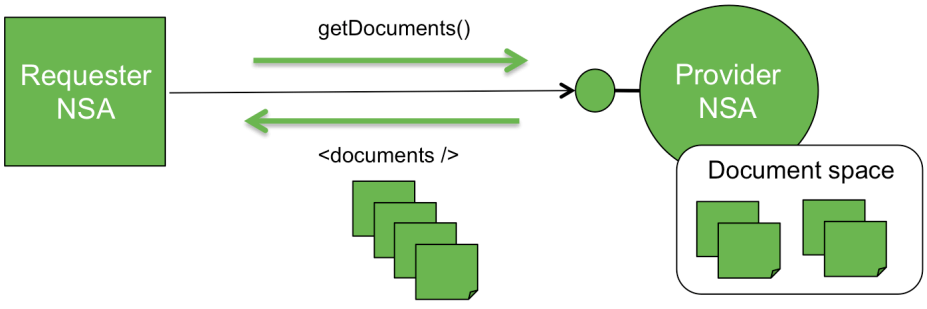

Figure 1 – Simple document get operation.

Figure 1 shows the simple *getDocuments()* operation that is invoked by the requester on the provider NSA to retrieve a set of documents from the document space. These simple document operations follow the standard request/response model.

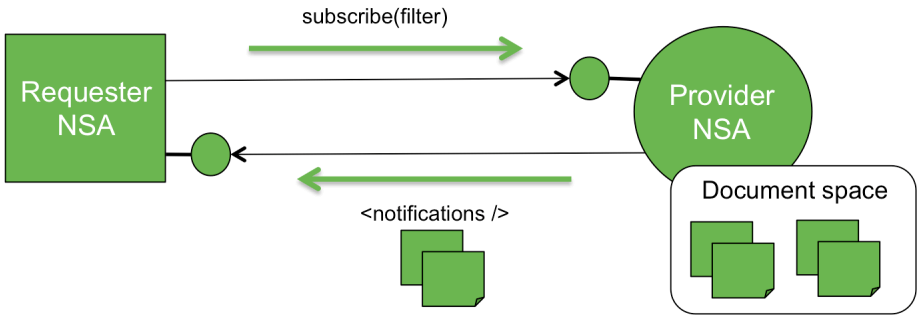

Figure 2 – Document change notification.

Figure 2 illustrates the interaction of the asynchronous publish/subscribe model supported by the discovery protocol's notification interface. In this example, the requester NSA requests a subscription supplying a filter to identify the documents of interest. In this subscription request the requester NSA also supplies a callback protocol endpoint that will receive the notifications delivered from the provider NSA. When there is a document event matching the subscription filter, the provider NSA will deliver the document to the requester NSA using the callback endpoint.

In Figure 3, an example flow showing how a document updated on one NSA gets propagated throughout the space via NSA peering relationships, so that in the end, all peer NSA within the space have an accurate version of each document within the space. In this example, the requester issues an update (version 1.2) to a document sourced on NSA A by using the *updateDocument()* operation. NSA A updates the local document space with the new version of the document, and looks through its subscription list to see if there are any NSA interested in the document. In this case, NSA B has registered for events on all documents within NSA A. NSA A issues a notification to NSA B with the updated document version 1.2. Similarly, NSA B will update its local document space and issue update notifications to NSA C and D who are also registered with NSA B for events on all documents. In this example, NSA D will receive update notifications for document version 1.2 from both NSA B and NSA C, however, NSA D will see that the document version for the two different notifications is identical, and discard the duplicate. NSA D then issues a notification to NSA E, which has registered for events on all documents within NSA D. NSA E updates its local document space, and since there are no further NSA to update, the flow for this update completes. It is key to note that an NSA does not propagate a document notification event back to the NSA from which it was originally received, as this NSA would just discard the update.

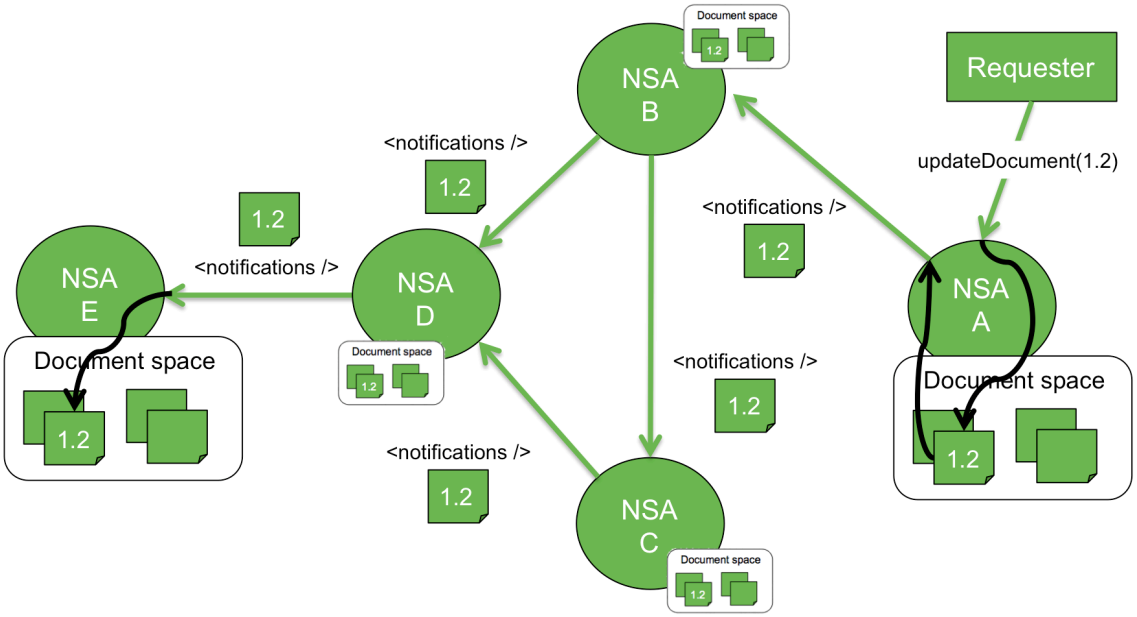

Figure 3 – Document propagation through space.

Additional operations, and more details on the document propagation mechanism are described in more detail in the coming sections.

2 NSI Service Framework

A basic overview of the functional components of the NSF architecture is described here to provide context to the reader. Addition detail can be found in [OGF NSF].

An NSA is said to be a requester if the NSA is capable of issue service requests, while it is a provider if it can receive service requests. An NSA may act as both a requester and a provider. The NSF defines three distinct roles for an NSA within the architecture:

- uRA: The ultimate Requester Agent is an NSA that originates but does not respond to service requests. The uRA could, for example, exist in a middleware application.
- uPA: The ultimate Provider Agent is an NSA that services requests by coordinating with the local Network Resource Manager (NRM) to manage network resources. The uPA responds to service requests, but never initiates them.
- AG: The Aggregator Agent (AG) is an NSA that has no physical network resources, but can orchestrate end-to-end network services on behalf of a user by utilizing the connection services exposed by an associated uPA or one or more child NSA. By definition the AG is both a requester and a provider NSA.

Figure 4 shows a pictorial representation of the three NSA roles within the NSF architecture.

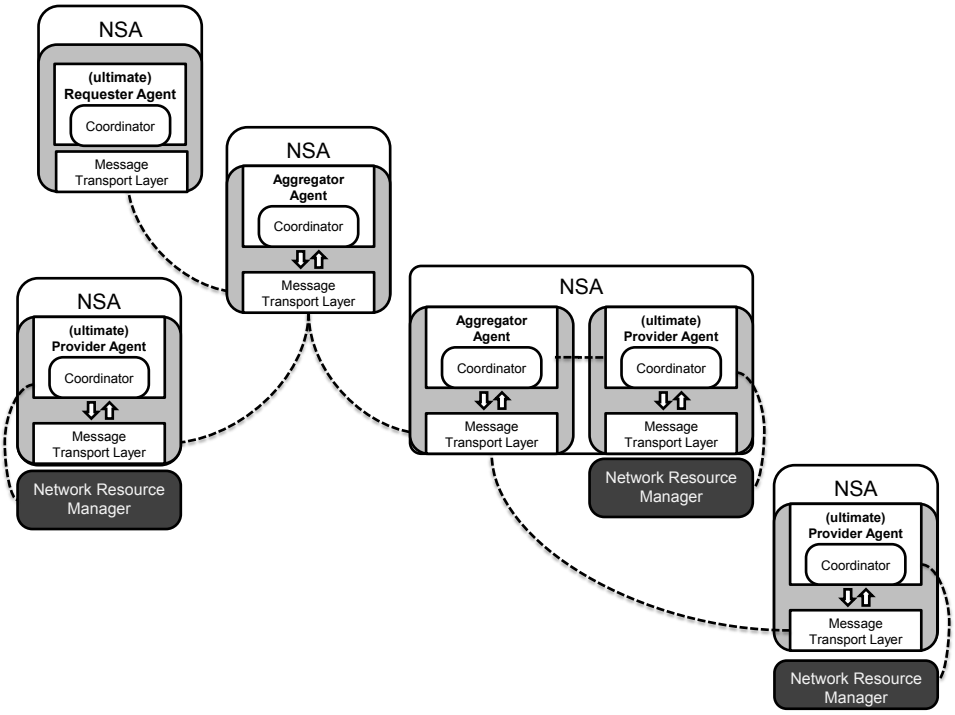

Figure 4 – Hierarchical NSA relationships.

An Aggregator NSA participating in the NSI Connection Service [OGF NSI-CS] requires access to a number of documents distributed by NSA through the NSI Discovery Service to perform basic functions such as:

- Bootstrapping communications with peer NSA (uRA, uPA, and other AG) using the NSA Discovery Document [OGF NSI-ND].
- Processing and validating connection service requests using NSI Service Definition Documents [OGF NSI-SD].
- Performing intelligent path finding for a requested connection service using NSI Topology Documents [OGF NSI-TS].

An ultimate Provider NSA participating in the NSI Connection Service does not require access to documents, but is required to distribute the following documents through the NSI Discovery Service:

- An NSA Discovery Document describing itself in detail, including supported interfaces, features, and networks.
- NSI Service Definition Documents for all services being offered by the local networks managed by the associated NRM.
- NSI Topology Documents of all advertised topology for the local networks managed by the associated NRM.

An ultimate Requester NSA participating in the NSI Connection Service does not produce any documents, however, it can optionally use the following documents from the NSI Discovery Service:

- The NSA Discovery Document from peer provider NSA to discover identity, supported interfaces, features, and networks.
- The NSI Service Definition Documents to determine available service types being offered within the network.
- The NSI Topology Documents if discovery of network ports or intelligent path finding is implemented by the uRA.

3 Documents

A document is any piece of information that needs to be distributed to all peers participating in the Discovery Service. A document is rapped in meta-data within the space to allow for identification and maintenance. The original document contents and associated meta-data are propagated untouched throughout the document space.

A document's meta-data entry is composed of the following attributes:

nsa The source NSA associated with the generation and management of the document within the network. This is assumed to be the NSA to which the document relates, however, there may be situations where this assumption is not true.

For example, if the document being generated is the NSA Discovery Document for NSA "*urn:ogf:network:example.com:2013:nsa:vixen*", then the *nsa* element should contain is the NSA identifier "*urn:ogf:network:example.com:2013:nsa:vixen*".

type The unique string identifying the type of this document. A document type is defined by the type and release of a data document. For example, NSI Topology version 1.0 and a NSI Topology version 2.0 would be considered two different document types:

- application/vnd.ogf.nsi.topology.v1+xml
- application/vnd.ogf.nsi.topology.v2+xml

The NSA Discovery Document 1.0 is defined as the type:

- application/vnd.ogf.nsi.nsa.v1+xml

id The identifier of the document. This value must be unique in the context of the nsa and type values.

version The version of the document, or more specifically, the date this version of the document was created. Any updates to the document must be tagged with a new version.

expires The date this version of the document expires and should be deleted from document space and any clients caching the document. More information is provided in Section 4.

signature An OPTIONAL digital signature of the document contents.

content The content of the document modeled by this document meta-data.

A document is uniquely identified by the tuple of NSA Identifier (*nsa*), Document Type (*type*), and Document Identifier (*id*). The Document Identifier need only be unique in the context of the NSA Identifier and Document Type. This allows for different types of documents to share the same identifier if they are considered directly related. It also implies that Document Identifiers do not need to be globally unique to be distributed or resolved in the document space.

Each meta-data entry contains a *version* attribute based on the date and time that version of the document was generated. As each new version is added to the space, it replaces the existing version and is propagated to all interested peers.

Meta-data also contains an *expires* attribute indicating when the document is no longer valid. Any clients caching a document that has expired MUST consider the information invalid and discard the document. An NSA within the space MAY keep the expired document for a period of time to guarantee all peers (both polling and subscriptions) have had time to receive the document after it has expired to cover the delete race condition described later in this document.

A document MAY also be digitally signed, generating a *signature* that can be associated with the document within the space. Clients of the space can use the *signature* to verify the originator and contents of the document. It is recommended that the document being signed includes the *identifier*, *version*, and *expires* meta-data attributes within the document itself so these values can also be verified if needed.

An NSA MUST not modify the contents of a document before propagating on to a peer unless that NSA is the owner of the document.

4 Time to Live

The Discovery Service uses the concept of Time To Live (TTL) to set an expiry date on documents exchanged through the protocol. There is no explicit delete operation within the protocol, so the TTL mechanism will ensure old documents eventually expire and are purged from the network. The three primary use cases for this feature are:

- An NSA has had a network removed from its configuration, resulting in the removal of a topology document; however, the associated topology document was previously announced into the network.

- An NSA has had a network name change, resulting in a new topology document being created and announced into the network; however, a topology document under the old network name was previously announced into the network and will not be refreshed when the new one is announced. When the TTL on the document is reached, all NSA holding a copy will purge it from the network.

- An NSA is removed from the network resulting in the removal of all associated topology documents; however, the associated topology documents were previously announced into the network. When the TTL on the document is reached, all NSA holding a copy will purge it from the network.

In all scenarios, when the TTL on the associated document is reached, all NSA holding a copy will purge it from the network. This will guarantee that the network will eventually return to an accurate state. In the case where the NSA knows a document should be deleted, it can perform an updated on the document, issuing a new version with an *expires* time set to a short period in the future. This update will propagate through the network and expire the document at the specified time instead of the original time,

An NSA MUST provide an *expires* time with each document published.

Enforcement of *expires* time MUST be based off of a network-synchronized clock.

The *expires* time SHOULD be a reasonable value computed based on the rate of expected change on the document.

An NSA MUST issue an updated document version to the network before the expiry time of the existing document. A reasonable lead-time should be provided to allow propagation of the new document throughout the network before the expiry of the existing version.

5 Subscriptions

To help support a more dynamic document distribution environment a publish/subscribe model is defined. Provider NSA allows requesters to subscribe to document events by specifying specific filters, that when matched, will generate document notifications to the subscriber. Requester can also publish documents into a specific provider's document space based on local security policies, which can then result in notification events to subscribed requesters if their registered filters match the event.

Each provider NSA also participates in the global document space as a requester, subscribing to document events on peer NSA for any document sourced by other NSA within the network. Through this subscription mechanism the provider NSA can dynamically build a global view of the document space without the need to perform document-polling operations on all peer NSA.

A subscription entry on a provider NSA is composed of the following attributes:

id	The provider assigned subscription identifier that uniquely identifies the subscription in the context of the provider.
version	The version of the subscription. Indicates the last time the subscription was modified by the requester.
requesterId	The identifier of the requester client that created the subscription. An NSA must use its unique NSA identifier for requesterId.
callback	The protocl endpoint on the requester that will receive the notifications delivered for this subscription.
filter	The OPTIONAL filter criteria to apply to document events to determine if a notification should be sent to the client.

Document events matching the supplied filter will generate notifications that will be delivered to the requester's protocol endpoint specified in the *callback* attribute. Only document events matching the filter criteria will generate a notification event to the subscriber. All other events will be discarded.

Subscription filters allow a subscriber to control the content delivered to their registered notification endpoint. A subscription request without a filter will result in a valid subscription that will match no document events. This can be used to create this initial subscription shell, which can later be modified to add filter criteria as needed.

The filter supports basic criteria:

include – Include notifications matching these criteria.

exclude - Exclude the notifications matching these criteria.

The include element specifies the document event match criteria to include, while the exclude element specifies those to specifically exclude. The include will be evaluated first, then the exclude will be applied. Each of the include and exclude criteria are composed of:

> *event* – The type of document event that will generate a notification. Currently only three events are supported (***All***, ***New***, ***Updated***). At least one of event criteria must be supplied. The default event criteria is ***All***.
>
> *or* – Any document matching any of the supplied *nsa*, document *type*, or document *id* values.
>
> *and* - Any document matching all of the supplied *nsa*, document *type*, or document *id* values.

The following filter subscribes for all document events (***All***) for all discovered documents:

```
<filter>
    <include>
        <event>All</event>
    </include>
</filter>
```

This previous filter would be the minimum filter criteria for an aggregator NSA. It would allow the aggregator to receive all document events from a peer NSA, building a complete view of documents discovered within the network. Multiple peers could deliver the same document events, however the aggregator could discard any duplicates. As the aggregator receives these duplicate events it may decide to modify the filter on the provider NSA issuing the duplicate events. The following filter is an example of a filter where the subscriber is still registered for all events, however, it has applied an exclude criteria to stop documents issued by NSA "*urn:ogf:network:example.com:2013:nsa:dasher*" from being sent to the subscriber endpoint:

```
<filter>
    <include>
        <event>All</event>
    </include>
    <exclude>
        <event>All</event>
        <or><nsa>urn:ogf:network:example.com:2013:nsa:dasher</nsa></or>
    </exclude>
</filter>
```

An alternative strategy for an aggregator is to initially subscribe to only new document events for its peers, expanding the filter by including individual documents, or documents from specific NSA in the filter as they are first discovered. Using this strategy, the subscribing NSA will only need to update a single subscription to start receiving document updates, instead of excluding from multiple peers as in the previous example.

The initial subscription filter subscribes for only new document events (***New***) for all discovered documents:

```
<filter>
    <include>
        <event>New</event>
    </include>
</filter>
```

As new document events arrive, the first peer to report the event can be the one who is configured to deliver future events for that document to the subscriber. The edited filter would still subscribe for all new document events (***New***), however, we add updates (***Updated***) document events for any documents provided by NSA "*urn:ogf:network:example.com:2013:nsa:vixen*":

```
<filter>
    <include>
        <event>New</event>
    </include>
    <include>
        <event>Updated</event>
        <event>Expired</event>
        <or><nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa></or>
    </include>
</filter>
```

Filtering on document type is also supported. The following filter subscribes for all document events (***All***) for discovered documents of type "application/vnd.ogf.nsi.discovery.v1":

```
<filter>
    <include>
        <event>All</event>
        <or><type>application/vnd.ogf.nsi.discovery.v1</type></or>
    </include>
</filter>
```

6 Operations

The logical operations supported by the NSI Discovery Service are classified into requester and provider interfaces, where a provider "provides" access to documents within the space, and a requester is "requesting" access to documents within the space. As described earlier, an NSA can participate in both the requester and provide roles of the protocol.

The provider interface for the NSI Discovery Service exposes the following logical operations:

getDocuments*([nsa], [type], [id], [lastDiscoveredTime])*

This operation returns a list of documents and the time of the latest document change on the provider NSA. If no filter parameters are supplied then all documents within the space will be returned. The following optional parameters can be supplied, and will be applied using logical AND:

nsa – The source NSA associated with the generation and management of the document within the network.

type - The unique string identifying the type of document to return.

id – The identifier of the document to return.

lastDiscoveredTime – Provides a time context to the provider requesting all documents that have been created or updated since the time specified in this parameter. This allows for an effective polling mechanism by using the latest document change time returned in the previous operation as a filter parameter in the next to retrieve only those documents that have been discovered (new or updated) since the last invocation of the API.

getLocalDocuments*([type], [id], [lastDiscoveredTime])*

This operation returns a list of documents associated with the queried provider NSA and the time of the latest document change on that provider NSA. If no filter parameters are supplied then all documents within the space will be returned. The following optional parameters can be supplied, and will be applied using logical AND:

type - The unique string identifying the type of document to return.

id – The identifier of the document to return.

lastDiscoveredTime – Provides a time context to the provider requesting all documents that have been created or updated since the time specified in this parameter. This allows for an effective polling mechanism by using the latest document change time returned in the previous operation as a filter parameter in the next to retrieve only those documents that have been discovered (new or updated) since the last invocation of the API.

getDocument*(nsa, type, id, [lastDiscoveredTime])*

This operation returns the requested document identified and the time of the latest change on the document. The following parameters are used to identify the specific document instance and are mandatory:

nsa – The source NSA associated with the generation and management of the document within the network.

type - The unique string identifying the type of document to return.

id – The identifier of the document to return.

If the optional filter parameter *lastDiscoveredTime* is provided, then the target document will only be returned if it has been updated since the time specified.

addDocument*(nsa, type, id, version, expires, [signature], contents)*

This operation adds a new document to the space associated with the provider NSA. Once the document has been successfully created on the provider, the provider will immediately send notifications to all subscriptions with filter criteria matching the document.

nsa – The source NSA associated with the generation and management of the document within the network.

type - The unique string identifying the type of this document.

id – The identifier of the document. This value must be unique in the context of the nsa and type values.

version - The version of the document, or more specifically, the date this version of the document was created.

expires - The date this version of the document expires and should be deleted from document space and any requesters caching the document.

signature - An OPTIONAL digital signature of the document contents.

contents - The contents of the document modeled by this document meta-data.

updateDocument*(nsa, type, id, version, expires, [signature], contents)*

This operation updates an existing document within the space associated with the provider NSA. A document can only be updated in the provider NSA acting as the source of the document. Any attempt to update a document from a provider other than the source of the document MUST be rejected. The operation returns a copy of the updated document.

This operation is also used to delete an existing document from the space associated with the provider NSA. For the delete of a document the requester issues a new document version with an *expire* time set to a reasonably short period in the future. This updated document propagates through the space to each NSA, updating the previous version to have the immediate expire time. All NSA receiving the document will then have an expired version.

nsa – The source NSA associated with the generation and management of the document within the network.

type - The unique string identifying the type of this document.

id – The identifier of the document. This value must be unique in the context of the nsa and type values.

version - The version of the document, or more specifically, the date this version of the document was created. Any updates to the document must be tagged with a new version.

expires - The date this version of the document expires and should be deleted from document space and any requesters caching the document.

signature - An OPTIONAL digital signature of the document contents.

contents - The contents of the document modeled by this document meta-data.

addSubscription*(requesterId, callback, filter)*

This operation subscribes a requester for document event notifications based on the supplied filter. Notifications will be delivered to the requester's protocol endpoint specified in the *callback* parameter. This operation returns the newly created subscription including the provider generated subscription *id*.

Once a subscription has been successfully created on the provider, the provider will immediately send notifications for all documents matching the filter criteria excluding the event filter (consider the event filter is set to ***All***). This allows a requester to initialize its local cache by getting a complete list of existing documents they are interested in monitoring. For example, if the event filter had been set to ***New*** for all documents, then this initialization behavior will send all matching documents as if they were just discovered.

requesterId - The identifier the requesting client would like to use for unique identification. An NSA must use its unique NSA identifier for requesterId.

callback – The requester's protocol endpoint that will receive the notifications delivered for this subscription.

filter - The filter criteria to apply to document events to determine if a notification should be sent to the client.

editSubscription*(id, requesterId, callback, filter)*

This operation allows a requester to edit an existing subscription. Once a subscription has been successfully edited on the provider, the provider will immediately send notifications for all documents matching the filter criteria excluding the event filter (consider the event filter is set to ***All***).

id – The provider assigned subscription identifier returned by the *addSubscription*() operation.

requesterId - The identifier the requesting client would like to use for unique identification. An NSA must use its unique NSA identifier for requesterId.

callback – The requester's protocol endpoint that will receive the notifications delivered for this subscription.

filter - The filter criteria to apply to document events to determine if a notification should be sent to the client.

deleteSubscription*(id)*

This operation deletes the subscription associated with *id* from the provider NSA. The deleted subscription is returned.

id – The provider assigned subscription identifier returned by the *addSubscription*() operation.

getSubscriptions*([requesterId], [lastDiscoveredTime])*

This operation returns a list of subscriptions and the time of the latest subscription change on the provider NSA. If no filter parameters are supplied then all subscriptions on the provider NSA will be returned. The following optional parameters can be supplied, and will be applied using logical AND:

requesterId – Return only subscriptions for this unique requester identifier.

lastDiscoveredTime – Provides a time context to the provider requesting all subscriptions that have been created or modified since the time specified in this parameter.

getSubscription*(id, [lastDiscoveredTime])*

This operation returns a single subscription identified by the *id* parameter and the time this subscription was last modified.

id – The provider assigned subscription identifier returned by the *addSubscription*() operation.

lastDiscoveredTime – OPTIONAL parameter provides a time context to the provider NSA requesting the subscription only be returned if it has been modified since the time specified in this parameter.

getAll*([lastDiscoveredTime])*

This operation returns a single subscription identified by the *id* parameter and the time this subscription was last modified.

id – The provider assigned subscription identifier returned by the *addSubscription*() operation.

lastDiscoveredTime – OPTIONAL parameter provides a time context to the provider NSA requesting the subscription only be returned if it has been modified since the time specified in this parameter.

7 NSA Bootstrap Procedure

One of the important uses of the NSI Discovery Service is the simplification of NSA provisioning through dynamic retrieval of the NSA Discovery Document. Utilizing the metadata contained in a peer NSA's Discovery Document it is possible to programmatically configure most of the information required to bring up the NSI suite of protocols. This section describes a basic procedure that can be followed that is compliant with the current NSI 2.0 protocol suite.

To bring up NSI communication between two peer NSA the NSA administrators must configure a local peering relationship:

1. Exchange TLS certificate and NSI Discovery Service endpoint with peer administrator.
2. Provision peer TLS certificate in NSA's local trust store to enable transport communications.
3. Provision peer certificate DN in NSA authorization module if additional application level validation is desired.
4. Provision the NSI Discovery Service URL in NSA for bootstrap procedure.

On NSA peering initialization:

1. The local NSA connects to Discovery Service on peer NSA using configured endpoint and TLS as a transport.
2. The local NSA performs a ***getLocalDocuments()*** operation to retrieve the peer NSA's Discovery Document and any other documents associated with the peer NSA.
3. The NSA identifier of the peer NSA and all associated networks is now known.
4. For each NSI service on local NSA, determine highest common interface version described in the peer NSA's Discovery Document. Both NSA should determine the same set of interface versions to use, however, the decision is made by the NSA behaving in the requester role.
5. Utilize interfaces and feature information as need.

For uRA (requester only NSA) this procedure is optional if the administrator would rather manually provision the required information.

8 Peer flooding and version sequencing

<Use Chin's example from topology presentation but make it for a generic document.>

9 Aggregator Algorithms

<Describe how an Aggregator would register with all peers, receive updates, and propagate notifications to other peers>

10 uPA Algorithms

<Describe how a uPA would service subscriptions and propagate notifications to subscribers. >

11 REST-based Protocol Profile

The NSI Discovery Service is implemented using a REST-based design pattern to create an HTTP based web service. This will allow for a lighter weight design, and simplify the overall protocol stack for a service that needs to be as simple as possible. This section provides a mapping from the abstract Discovery Service operations to concrete HTTP binding for the protocol. More information on the REST design pattern and best practices can be found in [FIELDING] and [RICH].

Table 1 describes the basic resources modeled in the Discovery Service REST API and the HTTP methods supported on the resources. As a standard design pattern, this protocol uses the HTTP GET method of retrieving and querying resources, the POST method for creating new instances of resources, the PUT method for updating a resource, and the DELETE method for deleting a resource.

Resource	Methods	Description
collection	GET	This root resource contains a collection of zero or more subscriptions and documents held within the NSA.
subscriptions	GET, POST	This resource represents a group of zero or more subscription instances.
subscription	GET, PUT, DELETE	This resource represents a single subscription instance.
documents	GET, POST	This resource represents a group of zero or more document instances.
document	GET, PUT, DELETE	This resource represents a single document instance.
local	GET	This resource represents a group of zero or more document instances associated with the local NSA.

Table 1 – Resources.

Table 2 describes the URI template mappings for the resources previously described.

Resource	URI	Description
collection	/	Using root URI with a GET operation will return a collection of zero or more subscriptions and documents held within the NSA.
subscriptions	/subscriptions	Using this URI with a GET operation will return a group of zero or more subscription instances. Using this URI with a POST operation will create a new subscription with the supplied criteria.
subscription	/subscriptions/{subscriptionId}	Use this URI template to access a single subscription instance based on subscription identifier. Using a GET operation will get the subscription identified by {*subscriptionId}*. Using a PUT operation will update the subscription identified by *{subscriptionId*} with the values supplied in the PUT body (*subscriptionRequest* element). Using a DELETE operation will remove the subscription identified by {*subscriptionId}*.
documents	/documents	Using this URI with a GET operation will return a group of zero or more document instances. Using this URI with a POST operation will create a new document with the supplied values (*document* element).
documents	/documents/{nsaId}	Use this URI template to access a list of document

		instances associated with an NSA identifier. Using this URI with a GET operation will return a group of zero or more document instances associated with the NSA identified by {*nsaId}*.
documents	/documents/{nsaId}/{type}	Use this URI template to access a list of document instances associated with an NSA identifier and specific document type. Using this URI with a GET operation will return a group of zero or more document instances of the document type *{type}* associated with the NSA identified by {*nsaId}*.
document	/documents/{nsaId}/{type}/{id}	Use this URI template to access a single document instance associated with an NSA identifier, document type, and document identifier. Using this URI with a GET operation will return a single document instance (*document* element) associated with the document identifier *{id},* the type *{type}, and* the NSA identified by {*nsaId}*. Using a PUT operation will update the document identified by *{id*} with the values supplied in the PUT body (*document* element). This can only be done by an authorized entity. Using a DELETE operation will remove the document identified by {*Id}*. This can only be done by an authorized entity.
local	/local	Using this URI with a GET operation will return a group of zero or more document instances associated with the local NSA.

Table 2 – URIs.

11.1 Content Encodings

The NSI Discovery Service Protocol mappings utilize custom MIME types carried in the *Content-Type* and *Accept* HTTP header parameters to identify the version of the resources carried in the HTTP body. It is design intension that the resources defined as part of the protocol are generic enough that they should not need to be up-versioned, however, in the case that the protocol needs to identify a change in format of the resource, a new MIME type will be created to identify the change.

On the HTTP POST and PUT request the *Content-Type* parameter identifies the version of resource carried in the body of the operation, and the *Accept* parameter identifies the version of resource acceptable on output. The HTTP response will contain a *Content-Type* parameter identifying the version of resource contained in the response.

The following string uniquely identifies this version of the discovery protocol:

"vnd.ogf.nsi.discovery.v1"

The following MIME type is defined to identify the XML content encoding for this specific version of the protocol:

"application/vnd.ogf.nsi.discovery.v1+xml"

The default content encoding for XML MUST also be supported for the newest version of the protocol:

"application/xml"

Further content encodings, including JSON, may be specified as needed.

11.2 Operations

This section describes the mappings of the abstract Discovery Service API operations to the physical REST-based protocol.

11.2.1 getDocuments

Method: GET /documents

This operation will return all document instances discovered within the document space, or a subset of documents based on supplied query parameters. Zero or more document instances will be returned in a *documents* element. Any results returned will be based on the permissions of the requester.

The URI template *"/documents/{nsa}/{type}"* can be used as an alternative to, or in conjunction with, the use of query parameters. Performing a GET on *"/documents/{nsa}/"* will return all documents associated with the specified NSA. Performing a GET on *"/documents/{nsa}/{type}"* will return all documents of *{type}* from the specified NSA.

Header Parameters

The following header parameters are supported for the documents resource.

Parameter	Value	Description
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.
If-Modified-Since	RFC1123 date string	Constrains the GET request to return only those documents that have been created or updated since the time specified in this parameter. If the query on the documents resource would have returned results, but applying these criteria results in an empty set of documents, a 304 (not modified) response will be returned without any message-body.

Query Parameters

The following query parameters are supported for the subscriptions resource. Query parameters are applied with a logical AND when there is more than one.

Parameter	Value	Description
id	String	Return all document resources containing the specified *Id*.
nsa	String	Return all document resources containing the specified *nsa* identifier. Cannot be used if the {nsa} URI component is provided.

type	String	Return all document resources containing the specified *type*. Cannot be used if the {type} URI component is provided.
summary	N/A	Will return summary results of any documents matching the query criteria. Summary results includes all document meta-data but not the *signature* or document *contents*.

Returns

The following information can be returned in response to the query.

Status Code	Element	Description
200	*documents*	Return *documents* element containing all document resources matching the query. If no documents match the query, then an empty *documents* element is returned.
304	N/A	Successful operation where there were no changes to any document resource given the *If-Modified-Since* criteria. Returns no message body.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***GET*** request on the "*/documents*" resource with a *type* query parameter. The result is a list of *document* resources matching the query parameter after any access control was applied:

```
GET /discovery/documents?type=application%2Fvnd.ogf.nsi.topology.v2+xml HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 648
Last-Modified: Mon, 10 Feb 2014 22:12:05 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:documents xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:document id="urn:ogf:network:example.com:2013:network:candycaneforest"
            version="2014-02-10T22:20:58Z" expires="2014-02-11T22:20:58Z">
        <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
        <type>application/vnd.ogf.nsi.topology.v2+xml</type>
        <signature>...</signature>
        <content>...</content>
    </tns:document>
    <tns:document id="urn:ogf:network:example.com:2013:network:lincolntunnel"
            version="2014-02-10T22:15:10Z" expires="2014-02-11T22:15:10Z">
        <nsa>urn:ogf:network:example.com:2013:nsa:prancer</nsa>
        <type>application/vnd.ogf.nsi.topology.v2+xml</type>
        <signature> ... </signature>
        <contents> ... </contents>
    </tns:document>
</tns:documents>
```

11.2.2 getLocalDocuments

Method: GET /local

A client can perform a GET operation on the special *"/local"* URI when it would like to discover all documents associated with the local NSA. The local NSA will return a *documents* element containing a list of zero or more document instances associated with the local NSA. This operation is equivalent to performing a GET operation on the URI *"/documents/{nsa}"*, however, for *"/local"* the client is not required to have previous knowledge of the local NSA identifier.

The URI template *"/local/{type}"* can be used as an alternative to, or in conjunction with, the use of query parameters. Performing a GET on *"/local/{type}/"* will return all documents of *{type}* associated with the local NSA.

Header Parameters

The following header parameters are supported for the documents resource.

Parameter	Value	Description
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.
If-Modified-Since	RFC1123 date string	Constrains the GET request to return only those documents that have been created or updated since the time specified in this parameter. If the query on the documents resource would have returned results, but applying these criteria results in an empty set of documents, a 304 (not modified) response will be returned without any message-body.

Query Parameters

The following query parameters are supported for the subscriptions resource. Query parameters are applied with a logical AND when there is more than one.

Parameter	Value	Description
id	String	Return all document resources containing the specified *Id*.
type	String	Return all document resources containing the specified *type*.
summary	N/A	Will return summary results of any documents matching the query criteria. Summary results includes all document meta-data but not the *signature* or document *contents*.

Returns

The following information can be returned in response to the query.

Status Code	Element	Description
200	*local*	Return *documents* element containing all document resources matching the query. If no documents match the query, then an empty *documents* element is returned.
304	NA	Successful operation where there were no changes to any document resource given the *If-Modified-Since* criteria.

		Returns no message body.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***GET*** request on the "*/local*" resource with a *type* query parameter. The result is a list of *document* resources matching the query parameter after any access control was applied:

```
GET /discovery/local?type=application%2Fvnd.ogf.nsi.topology.v2+xml HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 648
Last-Modified: Mon, 10 Feb 2014 22:12:05 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:local xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:document id="urn:ogf:network:example.com:2013:network:candycaneforest"
            version="2014-02-10T22:20:58Z" expires="2014-02-11T22:20:58Z">
        <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
        <type>application/vnd.ogf.nsi.topology.v2+xml</type>
        <signature>...</signature>
        <content>...</content>
    </tns:document>
    <tns:document id="urn:ogf:network:example.com:2013:network:lincolntunnel"
            version="2014-02-10T22:15:10Z" expires="2014-02-11T22:15:10Z">
        <nsa>urn:ogf:network:example.com:2013:nsa:prancer</nsa>
        <type>application/vnd.ogf.nsi.topology.v2+xml</type>
        <signature> ... </signature>
        <contents> ... </contents>
    </tns:document>
</tns:local>
```

11.2.3 addDocument

Method: POST /documents

The POST operation on the *"/documents*" resource will create a new document using the information supplied in the *document* element contained in the POST body. A successful operation will return the new document resource. This operation has restricted access for clients and is made available by the provider based on access control permissions.

Once a document has been successfully created on the provider, the provider will immediately send notifications to all subscriptions with filter criteria matching the document.

Header Parameters

The following header parameters are supported for the request for a new document resource.

Parameter	Value	Description
Content-Type	String	Identifies the content type encoding of the POST body contents. Must be a content type supported by the protocol.

Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

Body Parameters

The POST request must contain the *document* element containing the parameters of the *document* resource to be created.

Parameter	Value	Description
id	xsd:string	The identifier of the document. This value must be unique in the context of the nsa and type values.
version	xsd:dateTime	The version of the document. Typically the date this version of the document was created. Any updates to the document must be tagged with a new version.
expires	xsd:dateTime	The date this version of the document expires and should be deleted from the NSA (document server) and any clients caching the document.
nsa	xsd:anyURI	The source NSA associated with the generation and management of the document within the network.
type	xsd:string	The unique string identifying the type of this document.
signature	HolderType	The OPTIONAL digital signature of the document contents.
contents	HolderType	The contents of the document modeled by this document resource.

Returns

The following information can be returned in response to the POST.

Status Code	Element	Description
201	*document*	Returns a copy of the new document resource created as the result of a successful operation. The HTTP *Location* header field will contain the direct URI reference of the new document resource. It will be structured using the URI template $root/documents/{nsa}/{type}/{id}.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
403	*error*	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. An *error* element will be included populated with appropriate error information.
409	*error*	A document already exists with the same name (nsa/type/id). An update of an existing document should use the PUT operation.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***POST*** request on the "*/documents*" resource:

```
POST /discovery/documents HTTP/1.1
```

```
Accept: application/vnd.ogf.nsi.discovery.v1+xml
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:document xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="urn:ogf:network:example.com:2013:network:candycaneforest"
    version="2014-02-10T22:20:58Z" expires="2014-02-11T22:20:58Z">
    <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
    <type>application/vnd.ogf.nsi.topology.v2+xml</type>
    <signature>...</signature>
    <content>...</content>
</tns:document>

HTTP/1.1 201 Created
Date: Mon, 10 Feb 2014 22:21:59 GMT
Content-Length: 563
Last-Modified: Mon, 10 Feb 2014 22:21:58 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
Location:
/discovery/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.
topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest
<?xml version="1.0" encoding="UTF-8"?>
<tns:document xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="urn:ogf:network:example.com:2013:network:candycaneforest"
    version="2014-02-10T22:20:58Z" expires="2014-02-11T22:20:58Z">
    <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
    <type>application/vnd.ogf.nsi.topology.v2+xml</type>
    <signature>...</signature>
    <content>...</content>
</tns:document>
```

11.2.4 getDocument

Method: GET /documents/{nsa}/{type}/{id}

This operation will return a specific document instance discovered within the document space based on the URI template *"/documents/{nsa}/{type}/{id}",* where *{nsa*} is the NSA sourcing the document, *{type}* is the type of document, and *{id}* is the identifier of the specific document. The matching document is returned in a single *document* element.

Header Parameters

The following header parameters are supported for the subscriptions resource.

Parameter	Value	Description
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.
If-Modified-Since	RFC1123 date string	Constrains the GET request to return the matching document only if it has been updated since the time specified in this parameter. If the subscription resource does not meet these criteria, a 304 (not modified) response will be returned without any message-body.

Query Parameters

None.

Returns

The following information can be returned in response to the GET of a subscription.

Status Code	Element	Description
200	*document*	Successful operation returns the document identified by *{nsa}/{type}/{id}* in a *document* element. The *Last-Modified* header parameter will contain the time this document resource was last discovered.
304	NA	Successful operation where there were no changes to the document resource given the *If-Modified-Since* criteria. Returns no message body.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
404	*error*	Returned if the requested document was not found. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***GET*** request on the document resource identified by *the* URI *"/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest".* The result is a single *document* resource:

```
GET
/discovery/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.
topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:21:59 GMT
Content-Length: 563
Last-Modified: Mon, 10 Feb 2014 22:21:58 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:document xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="urn:ogf:network:example.com:2013:network:candycaneforest"
    version="2014-02-10T22:20:58Z" expires="2014-02-11T22:20:58Z">
    <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
    <type>application/vnd.ogf.nsi.topology.v2+xml</type>
    <signature>...</signature>
    <content>...</content>
</tns:document>
```

11.2.5 updateDocument

Method: PUT /documents/{nsa}/{type}/{id}

The PUT operation on the *"/documents/{nsa}/{type}/{id}*" resource will allow a client to edit the document corresponding to the identifier *{id},* using the information supplied in the *document* element contained in the PUT body. A successful operation will return the modified document and trigger any associated notifications within the NSA.

A document is deleted from the document space by updating it's expire date to a reasonably short period in the future. This updated document will get propagated throughout the document space and then expire, removing it from the space.

Header Parameters

The following header parameters are supported for the request edit a document resource.

Parameter	Value	Description
Content-Type	String	Identifies the content type encoding of the PUT body contents. Must be a content type supported by the protocol.
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

Body Parameters

The PUT request must contain the *document* element containing the existing parameters of the *document* resource if they were not modified, as well as any new/edited values.

Parameter	Value	Description
id	xsd:string	The identifier of the document. This value must be unique in the context of the nsa and type values.
version	xsd:dateTime	The version of the document. Typically the date this version of the document was created. Any updates to the document must be tagged with a new version.
expires	xsd:dateTime	The date this version of the document expires and should be deleted from the NSA (document server) and any clients caching the document.
nsa	xsd:anyURI	The source NSA associated with the generation and management of the document within the network.
type	xsd:string	The unique string identifying the type of this document.
signature	HolderType	The OPTIONAL digital signature of the document contents.
content	HolderType	The contents of the document modeled by this document resource.

Returns

The following information can be returned in response to the PUT.

Status Code	Element	Description
200	*document*	Returns a copy of the modified document resource as the result of a successful operation.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
403	*error*	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. An *error* element will be included populated with appropriate error information.
404	*error*	Returned if the requested document was not found. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the

processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***PUT*** request on the document *"/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest"* with updated version and expire attributes.

```
PUT
/discovery/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.
topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:document xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="urn:ogf:network:example.com:2013:network:candycaneforest"
    version="2014-02-12T22:20:58Z" expires="2014-02-13T22:20:58Z">
    <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
    <type>application/vnd.ogf.nsi.topology.v2+xml</type>
    <signature>...</signature>
    <content>...</content>
</tns:document>

HTTP/1.1 200 OK
Date: Mon, 12 Feb 2014 22:20:59 GMT
Content-Length: 563
Last-Modified: Mon, 12 Feb 2014 22:20:58 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
Location:
/discovery/documents/urn:ogf:network:example.com:2013:nsa:vixen/application%2Fvnd.ogf.nsi.
topology.v2+xml/urn:ogf:network:example.com:2013:network:candycaneforest
<?xml version="1.0" encoding="UTF-8"?>
<tns:document xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="urn:ogf:network:example.com:2013:network:candycaneforest"
    version="2014-02-12T22:20:58Z" expires="2014-02-13T22:20:58Z">
    <nsa>urn:ogf:network:example.com:2013:nsa:vixen</nsa>
    <type>application/vnd.ogf.nsi.topology.v2+xml</type>
    <signature>...</signature>
    <content>...</content>
</tns:document>
```

11.2.6 getSubscriptions

Method: GET /subscriptions

Return a *subscriptions* element containing a list of zero or more subscription instances based on supplied parameters and permissions of the requester.

Header Parameters

The following header parameters are supported for the subscriptions resource.

Parameter	Value	Description
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.
If-Modified-Since	RFC1123 date string	Constrains the GET request to return only those subscriptions that have been created or updated since the time specified in this parameter.

	If the query on the subscriptions resource would have returned results, but applying these criteria results in an empty set of documents, a 304 (not modified) response will be returned without any message-body.

Query Parameters

The following query parameters are supported for the subscriptions resource.

Parameter	Value	Description
requesterId	String	Return all subscription resources containing the specified *requesterId*.

Returns

The following information can be returned in response to the query.

Status Code	Element	Description
200	*subscriptions*	Return all subscription resources matching the query in a *subscriptions* element. If no subscriptions match the query, then an empty *subscriptions* element is returned.
304	NA	Successful operation where there were no changes to any subscription resources matching the query filter given the *If-Modified-Since* criteria. Returns no message body.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***GET*** request on the "*/subscriptions*" resource with a *requesterId* query parameter. The result is a list of *subscription* resources matching the query parameter after any access control is applied:

```
GET /discovery/subscriptions?requesterId=urn:ogf:network:example.com:2013:nsa:dasher
HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 648
Last-Modified: Mon, 10 Feb 2014 22:12:05 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscriptions xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:subscription id="9e223d413578" href="/discovery/subscriptions/9e223d413578"
version="2014-02-10T22:12:05Z">
        <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
        <callback>http://dasher.example.com/discovery/callback</callback>
        <filter>
            <include>
                <event>All</event>
            </include>
        </filter>
```

```
    </tns:subscription>
</tns:subscriptions>
```

11.2.7 addSubscription

Method: POST /subscriptions

The POST operation on the *"/subscriptions*" resource will create a new subscription using the information supplied in the *subscriptionRequest* element contained in the POST body. A successful operation will return the new subscription.

Once a subscription has been successfully created on the server, the server will immediately send notifications for all documents matching the filter criteria independent of the event filter.

Header Parameters

The following header parameters are supported for the request for a new subscription resource.

Parameter	Value	Description
Content-Type	String	Identifies the content type encoding of the POST body contents. Must be a content type supported by the protocol.
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

Body Parameters

The POST request must contain the *subscriptionRequest* element containing the initial parameters of the *subscription* resource to be created.

Parameter	Value	Description
requesterId	xsd:string	The identifier the requesting client would like to use for unique identification. An NSA must use its unique NSA identifier for *requesterId*.
callback	xsd:anyURI	The HTTP endpoint on the client host that will receive the notifications delivered for this subscription.
filter	FilterType	The *filter* criteria to apply to document events to determine if a notification should be sent to the client.

Returns

The following information can be returned in response to the POST.

Status Code	Element	Description
201	*subscription*	Returns a copy of the new subscription resource created as the result of a successful operation. The HTTP *Location* header field will contain the URI of the new subscription resource.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
403	*error*	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be

		repeated. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***POST*** request on the "*/subscriptions*" resource:

```
POST /discovery/subscriptions HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscriptionRequest
 xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
    <callback>http://dasher.example.com/discovery/callback</callback>
    <filter>
        <include>
            <event>All</event>
        </include>
    </filter>
</tns:subscriptionRequest>

HTTP/1.1 201 Created
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 405
Last-Modified: Mon, 10 Feb 2014 22:12:05 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
Location: /discovery/subscriptions/9e223d413578
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscription id="9e223d413578" href="/discovery/subscriptions/9e223d413578"
version="2014-02-10T22:12:05Z">
    <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
    <callback>http://dasher.example.com/discovery/callback</callback>
    <filter>
        <include>
            <event>All</event>
        </include>
    </filter>
</tns:subscription>
```

11.2.8 getSubscription

Method: GET /subscriptions/{id}

Returns a *subscription* element containing the subscription instance identified by the *{id}* parameter of the subscription.

Header Parameters

The following header parameters are supported for the subscriptions resource.

Parameter	Value	Description
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.
If-Modified-Since	RFC1123 date string	Constrains the GET request to return the matching subscription only if it has been updated since the time specified in this parameter.

	If the subscription resource does not meet these criteria, a 304 (not modified) response will be returned without any message-body.

Query Parameters

None.

Returns

The following information can be returned in response to the GET of a subscription.

Status Code	Element	Description
200	*subscription*	Successful operation returns the subscription identified by *id* in a *subscription* element. The *Last-Modified* header parameter will contain the time this subscription resource was last modified.
304	NA	Successful operation where there were no changes to the subscription resource identified by *id* given the *If-Modified-Since* criteria. Returns no message body.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
404	*error*	Returned if the requested subscription was not found. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***GET*** request on the resource identified by *id="9e223d413578",* and URI "*/subscriptions/9e223d413578*". The result is a single *subscription* resource matching the specified *id*:

```
GET /discovery/subscriptions/9e223d413578 HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 405
Last-Modified: Mon, 10 Feb 2014 22:12:05 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscription id="9e223d413578" href="/discovery/subscriptions/9e223d413578"
version="2014-02-10T22:12:05Z">
    <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
    <callback>http://dasher.example.com/discovery/callback</callback>
    <filter>
        <include>
            <event>All</event>
        </include>
    </filter>
</tns:subscription>
```

11.2.9 editSubscription

Method: PUT /subscriptions/{id}

The PUT operation on the *"/subscriptions/{id}*" resource will allow a client to edit the subscription corresponding to the identifier *{id},* using the information supplied in the *subscriptionRequest* element contained in the PUT body. A successful operation will return the modified subscription.

Header Parameters

The following header parameters are supported for the update request for a subscription resource.

Parameter	Value	Description
Content-Type	String	Identifies the content type encoding of the PUT body contents. Must be a content type supported by the protocol.
Accept	String	Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

Body Parameters

The PUT request must contain the *subscriptionRequest* element containing the existing parameters of the *subscription* resource if they were not modified, as well as any new/edited values. For example, if the filter parameter is being edited, then the *requesterId* and *callback* URI must be supplied with their existing values.

Parameter	Value	Description
requesterId	xsd:string	The identifier the requesting client would like to use for unique identification. An NSA must use its unique NSA identifier for *requesterId*.
callback	xsd:anyURI	The HTTP endpoint on the client host that will receive the notifications delivered for this subscription.
filter	FilterType	The *filter* criteria to apply to document events to determine if a notification should be sent to the client.

Returns

The following information can be returned in response to the PUT.

Status Code	Element	Description
200	*subscription*	Returns a copy of the modified subscription resource as the result of a successful operation.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
403	*error*	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. An *error* element will be included populated with appropriate error information.
404	*error*	Returned if the requested subscription was not found. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***PUT*** request on the "*/subscription/9e223d413578*" resource, editing the *filter* to include a new Updated event for the NSA "dasher". Notice that only those parameters that can be edited are included. In addition, the updated subscription resource will have a new version number corresponding to this update.

```
PUT /discovery/subscriptions/9e223d413578 HTTP/1.1
Accept: application/vnd.ogf.nsi.discovery.v1+xml
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscriptionRequest
 xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
    <callback>http://dasher.example.com/discovery/callback</callback>
    <filter>
        <include>
            <event>New</event>
        </include>
        <include>
            <event>Updated</event>
            <or><nsa>urn:ogf:network:example.com:2013:nsa:prancer</nsa></or>
        </include>
    </filter>
</tns:subscriptionRequest>

HTTP/1.1 200 OK
Date: Mon, 10 Feb 2014 22:20:59 GMT
Content-Length: 556
Last-Modified: Mon, 10 Feb 2014 22:20:58 GMT
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:subscription id="9e223d413578" href="/discovery/subscriptions/9e223d413578"
version="2014-02-10T22:20:58Z">
    <requesterId>urn:ogf:network:example.com:2013:nsa:dasher</requesterId>
    <callback>http://dasher.example.com/discovery/callback</callback>
    <filter>
        <include>
            <event>All</event>
        </include>
        <include>
            <event>Updated</event>
            <or><nsa>urn:ogf:network:example.com:2013:nsa:prancer</nsa></or>
        </include>
    </filter>
</tns:subscription>
```

11.2.10 deleteSubscription

Method: DELETE /subscriptions/{id}

Deletes the *subscription* resource identified by the *{id} URI* parameter if access control permissions allow the client to perform the delete operation on the target resource.

Header Parameters

None.

Query Parameters

None.

Returns

The following information can be returned in response to the DELETE of a subscription.

Status Code	Element	Description
204	*NA*	Successful delete operation returns no content.
400	*error*	Returned if a client specifies an invalid request. An *error* element will be included populated with appropriate error information.
403	*error*	Returned if the requested subscription was found, but the requesting client did not have permissions to delete the resource.
404	*error*	Returned if the requested subscription was not found. An *error* element will be included populated with appropriate error information.
500	*error*	Returned if an internal server error occurred during the processing of this request. An *error* element will be included populated with appropriate error information.

Example

The following example shows a valid ***DELETE*** request on the resource identified by *id="9e223d413578",* and URI "*/subscriptions/9e223d413578*". The result is a single *subscription* resource matching the specified *id*:

```
DELETE /discovery/subscriptions/9e223d413578 HTTP/1.1

HTTP/1.1 204 No Content
Date: Mon, 10 Feb 2014 22:12:59 GMT
```

11.2.11 Notifications

When a document event occurs matching a registered subscription the server must issue a *notification* to the client endpoint identified in the *subscription* resource. Multiple events can be grouped and delivered together in a single notification if these events occur within a reasonable period of time of each other. Notification delivery should not be delayed.

Notifications are also sent when a subscription is first created and will include any documents matching the initial filter criteria.

A failure in notification delivery may be the result of a temporary condition; so retrying notification delivery should be attempted for a reasonable period of time before discarding any pending notifications to a client and deleting the subscription. Notifications should not be discarded without deleting the subscription.

By creating a subscription, the client has entered a contractual agreement to expose an HTTP endpoint capable of receiving a POST operation with a message body containing a *notifications* element using the content encoding of the original subscription.

Method: POST <client supplied endpoint>

The POST operation on the *"<client supplied endpoint>*" will be a remote call from the discovery server holding the subscription to the client endpoint registered in the subscription. The client must return an HTTP 202 status code in response to the POST indicating it has successfully accepted the notification. Any other return code will result in a deletion of the subscription.

A server may periodically issue a POST to the client endpoint with a notification element containing zero elements. This should not be considered an error and the client must return an

HTTP 202 status code in response. The server to check the validity of a subscription can use this.

Header Parameters

The following header parameters are supported for the notification request to the client endpoint.

Parameter	Value	Description
Content-Type	String	Identifies the content type encoding of the POST body contents. Must be identical to the value as used by the client on subscription.

Body Parameters

The POST request must contain the *notifications* element, which will contain the list of zero or more notifications matching the subscription filter.

Parameter	Value	Description
providerId	xsd:anyURI	The identifier of the provider generating the notification. This is the provider on which the subscription was created.
id	xsd:string	The identifier of the subscription that generated the notifications.
href	xsd:anyURI	The URI reference for subscription that generated the notification. This can be used to directly access the subscription.
discovered	xsd:dateTime	The most recent document discovery time for the server in the context of when the notification was generated.
notification	NotificationListType	A list of zero or more notifications matching the subscription filter criteria.

Returns

The client receiving the notification must return an HTTP 202 status code in response to the POST. Any other status code will result in a deletion of the subscription.

Status Code	Element	Description
202	*NA*	Indicates the subscribed client has accepted the notification for processing.

Example

The following example shows a notification ***POST*** request on the "*/clientEndpoint*" resource:

```
POST /clientEndpoint HTTP/1.1
Content-Type: application/vnd.ogf.nsi.discovery.v1+xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:notifications xmlns:tns="http://schemas.ogf.org/nsi/2013/04/discovery/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    providerId="urn:ogf:network:example.com:2013:nsa:vixen"
    id="9e223d413578" href="/discovery/subscriptions/9e223d413578">
    <discovered>2014-02-10T22:20:58Z</discovered>
    <tns:notification>
        <discovered>2014-02-10T22:20:58Z</discovered>
        <event>New</event>
        <document id="urn:ogf:network:example.com:2013:network:lincolntunnel"
            version="2014-02-10T22:15:10Z" expires="2014-02-11T22:15:10Z">
            <nsa>urn:ogf:network:example.com:2013:nsa:prancer</nsa>
            <type>application/vnd.ogf.nsi.topology.v2+xml</type>
```

```
            <signature> ... </signature>
            <contents> ... </contents>
        </document>
    </tns:notification>
</tns:notifications>

HTTP/1.1 202 Accepted
Date: Mon, 10 Feb 2014 22:12:59 GMT
Content-Length: 0
```

12 Security Considerations

Documents carried by the NSI Discovery Service Protocol must be verifiable by requesters and providers within the network (e.g. the requester agent must be able to determine that the contents of the document was not altered during delivery, and is in fact, the same document published by the source provider). The NSI Discovery Service Protocol includes an element in the document meta-data to allow for the association of a digital signature by the publishing NSA, which can then be used by reach requester within the network to validate the authenticity of the attached document. Specification of the type of digital signature and algorithms used is left for definition outside of this specification since it may be document specific.

It is also assumed that exchange of documents between requester and provider NSA roles is secured to the level of other protocols within the NSI protocol suite. This security must include authentication, authorization, and confidentiality. To this end, the following security text is incorporated from [OGF NSI-CS].

TLS is used to ensure secure communication between requester and provider NSAs. TLS also supports X.509 certificates for authentication. Trust between NSAs is pairwise and MUST be established out-of-band. It is possible to have unidirectional trust between NSAs, i.e. reservations can only be created in one direction, as this is simply a policy special case. Transitive trust between NSAs cannot be assumed, i.e., NSAs A & B trust each other, and B & C trust each other, but this does not imply trust between A & C. However a request from A may end up using resources from C if passed through B. In the current security framework, B (if its policies permit) can proxy A's request to C. From C's point of view, it receives the request from B, and authenticates and authorizes the request using B's credentials. This document does not describe security policies, as these will always be site-specific. Note that due to the requirement for direct NSA-to-NSA communications (i.e. NSAs cannot forward communications via a third party NSA), message-level signing provides little value and is not used.

TLS provides message integrity, confidentiality and authentication via the X.509 certificates, and protects against replay attacks. Authorization is done at the NSAs application level. TLS version 1.0 MUST be supported. NSAs MAY use SSLv3 and TLS versions higher than 1.0 where possible.

13 Glossary

Aggregator NSA (Aggr)	The Aggregator NSA is a Provider Agent that acts as both a requester and provider NSA. It can service requests from other NSA, perform path finding, and distribute segment requests to child NSA for processing.
Connection Service (CS)	The NSI Connection Service is a service that allows an RA to request and manage a Connection from a PA. See [OGF NSI-CS].
Discovery Service	The NSI discovery service is a web service that allows the exchange of documents between RA and PA within the network. The NSA Discovery document is an example of information

	exchanged using this protocol.
Network Service Agent (NSA)	The Network Service Agent is a concrete piece of software that sends and receives NSI Messages. The NSA includes a set of capabilities that allow Network Services to be delivered.
Network Service Interface (NSI)	The NSI is the interface between RAs and PAs. The NSI defines a set of interactions or transactions between these NSAs to realize a Network Service.
Network Services Framework (NSF)	The Network Services framework describes an NSI message-based platform capable of supporting a suite of Network Services such as the Connection Service and the Topology Service. See [OGF NSF].
NSA Discovery document	The NSA Discovery document encapsulates descriptive meta-data associated with an NSA such as all NSI services and associated protocol interfaces offered by the NSA.
NSI Topology	The NSI Topology defines a standard ontology and a schema to describe network resources that are managed to create the NSI service. The NSI Topology as used by the NSI CS (and in future other NSI services) is described in [OGF NSI-TOP].
Requester/Provider Agent (RA/PA)	An NSA acts in one of two possible roles relative to a particular instance of an NSI. When an NSA requests a service, it is called a Requester Agent (RA). When an NSA realizes a service, it is called a Provider Agent (PA). A particular NSA may act in different roles at different interfaces.
NSI Service Definition	A document describing the service offered by an NSA and it's underlying network. A network can offer multiple services, and therefore, have multiple Service Definitions defined.
Simple Object Access Protocol (SOAP)	SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.
Ultimate PA (uPA)	The ultimate PA is a Provider Agent that has an associated NRM.
Ultimate RA (uRA)	The Ultimate RA is a Requester Agent is the originator of a service request.
XML Schema Definition (XSD)	XSD is a schema language for XML. See [W3C XSD]
eXtensible Markup Language (XML)	XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

14 Contributors

John H. MacAuley, ESnet, macauley@es.net

15 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.

16 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.

17 Full Copyright Notice

18 References

[BRADNER] Scott Bradner. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. The Internet Society. March 1997. http://tools.ietf.org/html/rfc2026

[RFC 6350] Simon Perreault. vCard Format Specification RFC 6350 (Standards Track), August 2011. URL http://tools.ietf.org/html/rfc6350.

[RFC 6351] S. Perreault. xCard: vCard XML Representation RFC 6351 (Standards Track), August 2011. URL http://tools.ietf.org/html/rfc6351.

[OGF NSF] Guy Roberts, et al. "OGF Network Service Framework v2.0", Group Working Draft (GWD), candidate Recommendation Proposed (R-P), January 28, 2014.

[OGF NSI-CS] Guy Roberts, et al. "OGF NSI Connection Service v2.0", Group Working Draft (GWD), candidate Recommendation Proposed (R-P), January 12, 2014.

[OGF NSI-TS] Jeroen van der Ham, GWD-R-P Network Service Interface Topology Representation, Group Working Draft (GWD), candidate Recommendations Proposed (R-P), January 2013.

[OGF NSI-ND] John MacAuley, et al. "Network Service Interface NSA Discovery Document v1.0", Group Working Draft (GWD), candidate Recommendation Proposed (R-P), February 18, 2014.

[OGF NSI-DS] John MacAuley, et al. “Network Service Interface Discovery Protocol v1.0”, Group Working Draft (GWD), candidate Recommendation Proposed (R-P), February 18, 2014.

[OGF NML] OGF GFD.206: Network Markup Language Base Schema version 1, http://www.gridforum.org/documents/GFD.206.pdf

[W3C XSD] W3C XML “Schema Definition Language (XSD) 1.1 Part 2: Datatypes”, http://www.w3.org/TR/xmlschema11-2/#anyURI

[FIELDING] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. UNIVERSITY OF CALIFORNIA, IRVINE, 2000, Chapter 5.

[RICH] L. Richardson, et al. Restful Web Services. O'Reilly Media; First Edition, May 15, 2007.

19 Appendix I – NSI Discovery Service Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
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.

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.

Copyright (C) Open Grid Forum (2009-2012). 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.

Open Grid Forum NSI Discovery Service Protocol v1.0.

Description: This is the NSI Discovery Service Protocol types schema for the
reference web services implementation of the OGF NSI Discovery Service v1.0.
Comments and questions can be directed to the mailing list group mailing list
(nsi-wg@ogf.org).
```

```
-->
<xsd:schema targetNamespace="http://schemas.ogf.org/nsi/2014/02/discovery/types"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://schemas.ogf.org/nsi/2014/02/discovery/types"
    xmlns:sig="http://www.w3.org/2000/09/xmldsig#"
    version="1.0">

    <xsd:annotation>
        <xsd:appinfo>ogf_nsi_discovery_protocol_v1_0.xsd 2014-02-20</xsd:appinfo>
        <xsd:documentation xml:lang="en">
            This is an XML schema document describing the OGF NSI Discovery
            Service Protocol v1.0.
        </xsd:documentation>
    </xsd:annotation>

    <!-- Import additional standard name spaces. -->
    <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="xmldsig-
core-schema.xsd"/>

    <!-- Collection for root resource definition. -->
    <xsd:element name="collection" type="tns:CollectionType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This root resource contains a collection of zero or more
                subscriptions and documents held within the NSA.

                HTTP operations: GET
                URI: /

                HTTP Parameters:
                  Accept - Identifies the content type encoding requested for
                  the returned results. Must be a content type supported by the
                  protocol.

                  If-Modified-Since - Return only entries discovered or
                    modified since this time.

                Query Parameters: None

                Returns (code, element):
                    200       collection
                        Return collection element containing all subscription
                        and document resources matching the query.  If no
                        subscriptions or documents match the query, then an empty
                        documents collection is returned.

                    304       None
                        Successful operation where there were no changes to any
                        subscription or document resource given the If-Modified-Since
                        criteria.  Returns no message body.

                    400       error
                        Returned if a client specifies an invalid request.  An
                        error element will be included populated with appropriate
                        error information.

                    500       error
                        Returned if an internal server error occurred during the
                        processing of this request. An error element will be
                        included populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="CollectionType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Type definition for a collection of discoverable resources.
                This type contains a list of subscriptions and docuemnts
```

```
                matching the query parameters.  Extensibility is added to
                allow inclusion of resources from other namespaces as needed.

                Elements:

                subscriptions - A list of subscription resources within the
                system.

                documents - A list of document resources stored within the
                document space of this provider.

                local - A list of document resources published by the local
                provider.

                other - Provides a flexible mechanism allowing additional elements
                to be provided from other namespaces without needing to update
                this schema definition.

                Attributes:

                other - Provides a flexible mechanism allowing additional attributes
                to be provided from other namespaces without needing to update
                this schema definition.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="tns:subscriptions" minOccurs="0" />
            <xsd:element ref="tns:documents" minOccurs="0" />
            <xsd:element ref="tns:local" minOccurs="0" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <!-- A list of subscriptions. -->
    <xsd:element name="subscriptions" type="tns:SubscriptionListType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The subscriptions resource contains a collection of zero or
                more subscriptions held within the provider NSA.

                HTTP operations: GET
                URI: /subscriptions

                HTTP Parameters:
                  Accept - Identifies the content type encoding requested for
                  the returned results. Must be a content type supported by the
                  protocol.

                  If-Modified-Since - Constrains the GET request to return only
                  those subscriptions that have been created or updated since the
                  time specified in this parameter.

                Query Parameters:
                  requesterId - Return all subscription resources containing the
                  specified requesterId.

                Returns (code, element):

                  200  subscriptions
                        Return all subscription resources matching the query in a
                        subscriptions element.  If no subscriptions match the query,
                        then an empty subscriptions element is returned.

                  304   None
                        Successful operation where there were no changes to any
                        subscription resources matching the query filter given the
                        If-Modified-Since criteria. Returns no message body.
```

```
                400  error
                     Returned if a client specifies an invalid request. An error
                     element will be included populated with appropriate error
                     information.

                500  error
                     Returned if an internal server error occurred during the
                     processing of this request. An error element will be included
                     populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="SubscriptionListType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Type definition for a list of subscription resources.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="tns:subscription" minOccurs="0" maxOccurs="unbounded" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <!-- A signle subscription resource definition. -->
    <xsd:element name="subscription" type="tns:SubscriptionType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The subscription resource contains a single subscription from
                the provider NSA.

                HTTP operations: GET
                URI: /subscriptions/{id}
                        {id} is the unique subscription identifier.

                HTTP Parameters:
                Accept - Identifies the content type encoding requested for
                the returned results. Must be a content type supported by the
                protocol.

                If-Modified-Since - Constrains the GET request to return only
                the subscription if it has been updated since the time specified
                in this parameter.

                Query Parameters: None

                Returns (code, element):

                200   subscription
                    Successful operation returns the subscription identified by
                    id in a subscription element.  The Last-Modified header
                    parameter will contain the time this subscription resource
                    was last modified.

                304 None
                    Successful operation where there were no changes to the
                    subscription resource identified by id given the
                    If-Modified-Since criteria. Returns no message body.

                400   error
                    Returned if a client specifies an invalid request. An error
                    element will be included populated with appropriate error
                    information.

                404 error
```

```
                    Returned if the requested subscription was not found.  An
                    error element will be included populated with appropriate
                    error information.

                500   error
                    Returned if an internal server error occurred during the
                    processing of this request. An error element will be included
                    populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="SubscriptionType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type models the subscription resource.

                Elements:

                requesterId   - The identifier of the requester client that created
                the subscription.  An NSA must use its unique NSA identifier for
                requesterId.

                callback - The HTTP endpoint on the client host that will receive
                the notifications delivered for this subscription.

                filter - The filter criteria to apply to document events to determine
                if a notification should be sent to the client.

                other - Provides a flexible mechanism allowing additional elements
                to be provided from other namespaces without needing to update
                this schema definition.

                Attributes:

                id - The provider assigned subscription identifier.

                href - The direct URI reference to the resource.

                version - The verison of the subscription.  Indicates the last
                time the subscription was modified.

                other - Provides a flexible mechanism allowing additional attributes
                to be provided from other namespaces without needing to update
                this schema definition.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="requesterId" type="xsd:string" />
            <xsd:element name="callback"    type="xsd:anyURI" />
            <xsd:element name="filter"      type="tns:FilterType" minOccurs="0" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="id"      use="required" type="xsd:string" />
        <xsd:attribute name="href"    use="required" type="xsd:anyURI" />
        <xsd:attribute name="version" use="required" type="xsd:dateTime" />
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <xsd:element name="subscriptionRequest" type="tns:SubscriptionRequestType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The subscriptionRequest is a collection of parameters from the
                subscription resource that is used to create a new subscription
                resource or update an existing subscription resource.

                Once a subscription has been successfully created or updated on
                the provider the server will immediately send notifications for
```

```
                all documents matching the filter criteria independent of the
                event filter.

                HTTP operations: POST (create), PUT (update)
                URI: /subscriptions

                HTTP Parameters:
                Content-Type - Identifies the content type encoding of the POST
                body contents.  Must be a content type supported by the protocol.

                Accept - Identifies the content type encoding requested for
                the returned results. Must be a content type supported by the
                protocol.

                If-Modified-Since - Constrains the GET request to return only
                the subscription if it has been updated since the time specified
                in this parameter.

                Query Parameters: N/A

                Returns (code, element):

                201   subscription
                    Returns a copy of the new subscription resource created as
                    the result of a successful operation.  The HTTP Location
                    header field will contain the URI of the new subscription
                    resource.

                400   error
                    Returned if a client specifies an invalid request. An error
                    element will be included populated with appropriate error
                    information.

                403   error
                    The server understood the request, but is refusing to fulfill
                    it. Authorization will not help and the request SHOULD NOT be
                    repeated.  An error element will be included populated with
                    appropriate error information.

                500   error
                    Returned if an internal server error occurred during the
                    processing of this request. An error element will be included
                    populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="SubscriptionRequestType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type models a subset of parameters from the subscription
                resource used during creation and updates.

                Elements:

                requesterId   - The identifier the requesting client would like to
                use for unique identification.  An NSA must use its unique NSA
                identifier for requesterId.

                callback - The HTTP endpoint on the client host that will receive
                the notifications delivered for this subscription.

                filter - The filter criteria to apply to document events to determine
                if a notification should be sent to the client.

                other - Provides a flexible mechanism allowing additional elements
                to be provided from other namespaces without needing to update
                this schema definition.
```

```
                Attributes:

                other - Provides a flexible mechanism allowing additional attributes
                to be provided from other namespaces without needing to update
                this schema definition.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="requesterId" type="xsd:string" />
            <xsd:element name="callback"    type="xsd:anyURI" />
            <xsd:element name="filter"      type="tns:FilterType" minOccurs="0" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <xsd:complexType name="FilterType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type is the base notification filter for subscriptions.
                The include element specifies the document event match criteria
                to include, while the exclude element specifies those to
                specifically exclude.  The include will be evaluated first, then
                the exclude will be applied.

                Elements:

                include - Include notifications matching these criteria.

                exclude - Exclude the notifications matching these criteria.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="include" type="tns:FilterCriteriaType" minOccurs="0"
maxOccurs="unbounded" />
            <xsd:element name="exclude" type="tns:FilterCriteriaType" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="FilterCriteriaType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type models the criteria that can be included in the
                notfication filter for subscriptions.

                Elements:

                event - The type of document event that will generate a
                notification.  Currently only three events are supported (All,
                New, Updated).  At least one of event criteria must be
                supplied.  The default event criteria is All.

                or - Any document matching any of the supplied nsa, document
                type, or document id values.

                and - Any document matching all of the supplied nsa, document
                type, or document id values (logical AND).
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="event" type="tns:DocumentEventType" default="All"
minOccurs="1" maxOccurs="3" />
            <xsd:element name="or"    type="tns:FilterOrType" minOccurs="0"
maxOccurs="unbounded" />
            <xsd:element name="and"   type="tns:FilterAndType" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
```

```
    </xsd:complexType>

    <xsd:simpleType name="DocumentEventType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This is a simple string type enumerating the types of document
                events that can be included in a filter.

                All - Matches all document events.

                New - Matches new documents that are discovered in the space.

                Updated - Matches existing documents in the space that are updated.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="All"/>
            <xsd:enumeration value="New"/>
            <xsd:enumeration value="Updated"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="FilterAndType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This filter criteria type lists elements that can be matched in a
                document as part of the decision to generate or not generate a
                notification.  The supplied nsa, document type, and document id
                values are evaluted as a logical AND so that all included values
                must match.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="nsa"  type="xsd:anyURI" minOccurs="0" />
            <xsd:element name="type" type="xsd:string" minOccurs="0" />
            <xsd:element name="id"   type="xsd:string" minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="FilterOrType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This filter criteria type lists elements that can be matched in a
                document as part of the decision to generate or not generate a
                notification.  The supplied nsa, document type, and document id
                values are evaluted as a logical OR so that any included values
                that match result in a criteria match.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:choice maxOccurs="unbounded">
                <xsd:element name="nsa"  type="xsd:anyURI" />
                <xsd:element name="type" type="xsd:string" />
                <xsd:element name="id"   type="xsd:string" />
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>

    <!-- A list of notifications. -->
    <xsd:element name="notifications" type="tns:NotificationListType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                When a document event occurs matching a registered subscription
                the provider must issue a notification to the requester endpoint
                identified in the subscription resource.  This element is sent
                in the body of a POST request to the requester endpoint.

                Multiple events can be grouped and delivered together in a single
                notification if these events occur within a reasonable period of
```

```
            time of each other. Notification delivery should not be delayed.

            Notifications are also sent when a subscription is first created,
            and after a subscription is modified.  This notification will
            include any documents matching the filter criteria.

            HTTP operations: POST
            URI: /client-supplied-endpoint

            HTTP Parameters:

            Content-Type - Identifies the content type encoding of the POST
            body contents.  Must be identical to the value as used by the
            client on subscription.

            Query Parameters: N/A

            Returns (code, element):

            202    None
                Indicates the subscribed client has accepted the notification
                for processing.  The client receiving the notification must
                return an HTTP 202 status code in response to the POST.
                Any other status code will result in a deletion of the
                subscription.
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:complexType name="NotificationListType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type definition for a list of notifications.

            Elements:

            discovered - The most recent document discovery time for the
            provider in the context of when the notification was generated.

            notification - A list of zero or more notifications matching the
            subscription filter criteria.

            Attributes:

            providerId - The identifier of the provider generating the
            notification.  This is the provider on which the subscription
            was created.

            id - The identifier of the subscription that generated the
            notifications.

            href - The URI reference for subscription that generated the
            notification.  This can be used to directly access the
            subscription.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="discovered" type="xsd:dateTime" />
        <xsd:element ref="tns:notification" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="providerId" use="required" type="xsd:anyURI" />
    <xsd:attribute name="id"         use="required" type="xsd:string" />
    <xsd:attribute name="href"       use="required" type="xsd:anyURI" />
</xsd:complexType>

<!-- A single notfication. -->
<xsd:element name="notification" type="tns:NotificationType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
```

```
                This element models a single document notification and is
                included in the notifications element.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="NotificationType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type models a single document notification event.

                Elements:

                discovered - The time this document event was detected on the
                provider.  It is not the time the notification was generated.
                It also should be noted that this time could be a considerable
                period in the past if the notification was sent as the result
                of a subscription creation or edit.

                event - The type of document event this notification represents.

                document - The document metadata entry associated with the
                notification.

                other - Provides a flexible mechanism allowing additional element
                to be provided from other namespaces without needing to update
                this schema definition.

                Attributes:

                other - Provides a flexible mechanism allowing additional attributes
                to be provided from other namespaces without needing to update
                this schema definition.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="discovered" type="xsd:dateTime" />
            <xsd:element name="event" type="tns:DocumentEventType" />
            <xsd:element name="document"   type="tns:DocumentType" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <!-- A list of documents. -->
    <xsd:element name="documents" type="tns:DocumentListType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The documents element models a list of documents from the
                document space.

                HTTP operations: GET
                URI: /documents/{nsa}/{type}

                The documents element contains document resources discovered
                within the document space, or a subset of documents based on
                supplied query parameters.  Zero or more document instances will
                be returned in a documents element.

                The URI template "/documents/{nsa}/{type}" can be used as an
                alternative to, or in conjunction with, the use of query
                parameters.  Performing a GET on "/documents/{nsa}/" will
                return all documents associated with the specified NSA.
                Performing a GET on "/documents/{nsa}/{type}" will return
                all documents of {type} from the specified NSA.

                HTTP Parameters:
```

Accept - Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

If-Modified-Since - Constrains the GET request to return only those documents that have been created or updated since the time specified in this parameter.

Query Parameters:

id (string) - Return all document resources containing the specified Id.

nsa (string) - Return all document resources containing the specified nsa identifier. Cannot be used if the {nsa} URI component is provided.

type (string) - Return all document resources containing the specified type. Cannot be used if the {type} URI component is provided.

summary (none) - Will return summary results of any documents matching the query criteria. Summary results includes all document meta-data but not the signature or document contents.

Returns (code, element):

200 documents
Return all document resources matching the query in a documents element. If no documents match the query, then an empty documents element is returned.

304 None
Successful operation where there were no changes to any subscription resources matching the query filter given the If-Modified-Since criteria. Returns no message body.

400 error
Returned if a client specifies an invalid request. An error element will be included populated with appropriate error information.

500 error
Returned if an internal server error occurred during the processing of this request. An error element will be included populated with appropriate error information.

HTTP operations: POST
URI: /documents

The POST operation on the "/documents" resource will create a new document using the information supplied in the document element contained in the POST body. A successful operation will return the new document resource. This operation has restricted access for clients and is made available by the server based on access control permissions.

Once a document has been successfully created on the server, the server will immediately send notifications to all subscriptions with filter criteria matching the document.

HTTP Parameters:

Content-Type - Identifies the content type encoding of the POST body contents. Must be a content type supported by the protocol.

Accept - Identifies the content type encoding requested for the returned results. Must be a content type supported by the protocol.

```
            If-Modified-Since - Constrains the GET request to return only
            those documents that have been created or updated since the
            time specified in this parameter.

            Body Parameters:

            document - The document to add to the document space of the
            local provider.

            Returns (code, element):

            201   document
                Returns a copy of the new document resource created as the
                result of a successful operation.  The HTTP Location header
                field will contain the direct URI reference of the new
                document resource.  It will be structured using the URI
                template $root/documents/{nsa}/{type}/{id}.

            400   error
                Returned if a client specifies an invalid request.  An error
                element will be included populated with appropriate error
                information.

            403   error
                The server understood the request, but is refusing to fulfill
                it. Authorization will not help and the request SHOULD NOT
                be repeated.  An error element will be included populated
                with appropriate error information.

            409   error
                A document already exists with the same name (nsa/type/id).
                An update of an existing document should use the PUT
                operation.

            500   error
                Returned if an internal server error occurred during the
                processing of this request. An error element will be
                included populated with appropriate error information.
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="local" type="tns:DocumentListType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            The local element models a list of documents from the document
            space published by the local provider NSA.

            HTTP operations: GET
            URI: /local/{type}

            The local element contains document resources published by the
            local provider, or a subset of documents based on supplied query
            parameters.  Zero or more document instances will be returned in
            a local element.

            A client can perform a GET operation on the special "/local" URI
            when it would like to discover all documents associated with the
            local provider NSA.  This operation is equivalent to performing a
            GET operation on the URI "/documents/{nsa}", however, for "/local"
            the client is not required to have previous knowledge of the
            provider NSA identifier.

            The URI template "/local/{type}" can be used as an alternative to,
            or in conjunction with, the use of query parameters.  Performing
            a GET on "/local/{type}/" will return all documents of {type}
            associated with the local NSA.

            HTTP Parameters:
```

```
                Accept - Identifies the content type encoding requested for
                the returned results. Must be a content type supported by the
                protocol.

                If-Modified-Since - Constrains the GET request to return only
                those documents that have been created or updated since the
                time specified in this parameter.

                Query Parameters:

                id (string) - Return all document resources containing the
                specified Id.

                type (string) - Return all document resources containing the
                specified type. Cannot be used if the {type} URI component is
                provided.

                summary       (none) - Will return summary results of any documents
                matching the query criteria.  Summary results includes all
                document meta-data but not the signature or document contents.

                Returns (code, element):

                200    local
                    Return all document resources matching the query in a
                    documents element.  If no documents match the query,
                    then an empty documents element is returned.

                304   None
                    Successful operation where there were no changes to any
                    document resources matching the query filter given the
                    If-Modified-Since criteria. Returns no message body.

                400    error
                    Returned if a client specifies an invalid request. An error
                    element will be included populated with appropriate error
                    information.

                500    error
                    Returned if an internal server error occurred during the
                    processing of this request. An error element will be included
                    populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="DocumentListType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type provides a list of zero or more documents.

                Elements:

                document - The document meta-data entry within the document space.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="tns:document" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>

    <!-- A single document. -->
    <xsd:element name="document" type="tns:DocumentType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The document element models the metadata for a single document
                from the document space.
```

```
HTTP operations: GET
URI: /documents/{nsa}/{type}/{id}

This operation will return a specific document instance
discovered within the document space based on the URI template
"/documents/{nsa}/{type}/{id}", where {nsa} is the NSA sourcing
the document, {type} is the type of document, and {id} is the
identifier of the specific document.  The matching document is
returned in a single document element.

HTTP Parameters:

Accept - Identifies the content type encoding requested for
the returned results. Must be a content type supported by the
protocol.

If-Modified-Since - Constrains the GET request to return only
those documents that have been created or updated since the
time specified in this parameter.

Query Parameters: None.

Returns (code, element):

200   local
    Successful operation returns the document identified by
    {nsa}/{type}/{id} in a document element.  The Last-Modified
    header parameter will contain the time this document resource
    was last discovered.

304   None
    Successful operation returns the document identified by
    {nsa}/{type}/{id} in a document element.  The Last-Modified
    header parameter will contain the time this document resource
    was last discovered.

400   error
    Returned if a client specifies an invalid request. An error
    element will be included populated with appropriate error
    information.

404   error
    Returned if the requested document was not found.  An error
    element will be included populated with appropriate error
    information.

500   error
    Returned if an internal server error occurred during the
    processing of this request. An error element will be included
    populated with appropriate error information.

HTTP operations: PUT
URI: /documents/{nsa}/{type}/{id}

The PUT operation on the "/documents/{nsa}/{type}/{id}" resource
will allow a client to edit the document corresponding to the
identifier {id}, using the information supplied in the document
element contained in the PUT body.  A successful operation will
return the modified document and trigger any associated
notifications within the NSA.

A document is deleted from the document space by updating it's
expire date to a reasonably short period in the future.  This
updated document will get propagated throughout the document
space and then expire, removing it from the space.

 HTTP Parameters:

 Content-Type - Identifies the content type encoding of the PUT
```

```
                body contents.  Must be a content type supported by the
                protocol.

                Accept - Identifies the content type encoding requested for
                the returned results. Must be a content type supported by the
                protocol.

                Body Parameters:

                document - The document to update in the document space of the
                local provider. The PUT request must contain the document
                element containing the existing parameters of the document
                resource if they were not modified, as well as any new/edited
                values.

                Returns (code, element):

                200 document
                   Returns a copy of the modified document resource as the
                   result of a successful operation.

                400 error
                   Returned if a client specifies an invalid request.  An
                   error element will be included populated with appropriate
                   error information.

                403 error
                   The server understood the request, but is refusing to fulfill
                   it. Authorization will not help and the request SHOULD NOT be
                   repeated.  An error element will be included populated with
                   appropriate error information.

                404 error
                   Returned if the requested document was not found.  An error
                   element will be included populated with appropriate error
                   information.

                500 error
                   Returned if an internal server error occurred during the
                   processing of this request. An error element will be included
                   populated with appropriate error information.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="DocumentType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The DocumentType type definition models a data relating to a
                single document exchanged within the network.  Meta-data
                associated with the document, document signature, and the
                document itself is encapsulated in this type.  The type
                itself is structured such that it does not need to be
                manipulated between receiving and propagating to a peer.

                A document is uniquely named within the network by the tuple
                of nsa, type, and id.  The identifier element itself does not
                need to be unique within the network; it must just be unique
                within the context of the nsa and type elements.  These rules
                allow the reuse of the same id value for a document of different
                types under the same source NSA.  This is important for both
                searching, and for associating the same naming attribute to
                related documents.

                An NSA must not modify the contents of a DocumentType before
                propagating on to a peer unless that NSA is the owner of the
                document.

                Elements:
```

```
                nsa    - The source NSA associated with the generation and management
                of the document within the network.

                type - The unique string identifying the type of this document.

                signature - The OPTIONAL digital signature of the document
                contents.

                content - The contents of the document modeled by this document
                resource.

                other - Provides a flexible mechanism allowing additional elements
                to be provided from other namespaces without needing to update
                this schema definition.

                Attributes:

                id - The identifier of the document.  This value must be unique
                in the context of the nsa and type values.

                version       - The version of the document.  Typically the date this
                version of the document was created.  Any updates to the document
                must be tagged with a new version.

                expires       - The date this version of the document expires and should
                be deleted from the NSA (document server) and any clients caching
                the document.

                other - Provides a flexible mechanism allowing additional attributes
                to be provided from other namespaces without needing to update
                this schema definition.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="nsa"       type="xsd:anyURI" />
            <xsd:element name="type"      type="xsd:string" />
            <xsd:element name="signature" type="tns:AnyType" minOccurs="0" />
            <xsd:element name="content"   type="tns:AnyType" minOccurs="0" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="id" use="required" type="xsd:string" />
        <xsd:attribute name="version" use="required" type="xsd:dateTime" />
        <xsd:attribute name="expires" use="required" type="xsd:dateTime" />
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <xsd:complexType name="AnyType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type is used to hold a document or signature within the
                document metadata.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <xsd:element name="error"     type="tns:ErrorType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The error element is returned in an HTTP response when an error
                has occured servicing the request on the provider.
            </xsd:documentation>
        </xsd:annotation>
```

```
    </xsd:element>

    <xsd:complexType name="ErrorType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type models errors returned from Discovery Service operations.

                Elements:

                code - The integer error code for the specific error.

                label - A character string label for the error.

                description - A detailed description of error.

                resource - The resource that caused the error.

                Attributes:

                id - The unique identifier of the error for correlation with logs.

                date - The date and time the error occured.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="code"        type="xsd:int" />
            <xsd:element name="label"       type="xsd:string" />
            <xsd:element name="description" type="xsd:string" />
            <xsd:element name="resource"    type="xsd:anyURI" />
        </xsd:sequence>
        <xsd:attribute name="id"   type="xsd:string" use="required" />
        <xsd:attribute name="date" type="xsd:dateTime" use="required" />
    </xsd:complexType>
</xsd:schema>
```