Guy Roberts, GÉANT
December 2, 2019

NSI Connection Service v2.0 to v2.1 Delta

Status of This Document

This document provides information to the Grid community on the NSI Connection Service v2.1 document.

Copyright Notice

Abstract

This document is an informational guide to the differences between the NSI Connection Service version 2.0 and version 2.1.

Contents

1. Schema changes in v2.1

The schema used in the NSI Connection Service v2.1 is backward compatible with v2.0 in terms of its behavior. This allows the V2.0 schema namespace to be carried over when upgrading to v2.1. The core schema from version 2.0 is retained, on top of this are a new set of optional schemas for:

- *resvFailed*
- *pathTrace*
- *ifModifiedSince*
- *lastModified*
- ERO exclusions

2. Overview of new features in v2.1

2.1 Resource availability feedback

This feature allows the RA to discover information about the resources available in the PA.

- Returns the resource availability in the *resvFailed* message
- Creates the optional *serviceException* variables feedback element
- String based, more verbose and rigid

2.2 pathTrace

- This attribute provides a trace of Networks that have been transited
- It is used to allow PAs to apply policy

2.3 *LastModified* attribute for queries

- Adds a timestamp on incident (i.e. state change on reservation)
- Enables a query to ask for events *ifModifiedSince* based on incident timestamps.
- Adds *ifModifiedSince* element to *QueryType*
- Adds *lastModified* element to *QuerySummaryConfirmedType*.

2.4 Error codes have been updated and moved to separate GWD document.

2.5 ERO exclusions have been added

- This allows the user to exclude resources from a path
- This is an optional extension to the existing point to point service specific schema

2.6 Additional various errata updates

- Clarifications and Minor behavioral changes

3. Specific Changes

This section lists all of the specific changes that appear for the first time in NSI CS v2.1. The changes are listed in order that they appear in the NSI CS v2.1 document.

3.1 Section 4.2 Explicit Routing Object

The following sentence has been added to the second paragraph of section 4.2:
"Also note that STP at either end of an SDP can be used to uniquely identify the SDP to transit. Both STPs in a single SDP are not required in the ERO, and in fact, only a single one should be specified."

3.2 Section 5.3.1 Reservation State Machine

The Reservation State Machine has been modified to allow the *rsvTimeout* to be propagated by an Aggregator. Figure 3 Reservation Sate Machine has been updated and the associated text has been revised to clarify the operation of the RSM.

The following text has been removed:

> The **reserveTimeout** state is only implemented where the ultimate Provider Agent functionality is present

The following text has been added to this section:

> (If a requester fails to commit a held reservation after a certain period of time, the provider times out the reservation and the held resources are released.) This triggers the transition to the **reserveTimeout** state within the ultimate Provider Agent, which in turn causes a reserve timeout notification to be sent upstream towards the requester. If the requester is an Aggregator Agent, it will transition to the **reserveTimeout** state upon receipt of the reserve timeout notification, and continue to forward the notification upstream. This transition to the **reserveTimeout** state by the Aggregator Agent allows it to reflect that one or more of its downstream ultimate Provider Agents have timed out a reservation.

The following text has also been added to this section:

> While a reservation is being modified the <*reservationState*> reflects the current state of the modification even though the <criteria> represents the last committed version.

3.3 Sections 5.3.1 through 5.3.3 Reservation State Machine

Figure 3: RSM, Figure 4 PSM and Figure 5 LSM have all been updated with the following clarification information in the diagram key: "Either *input event* or *input message* can trigger output (logical disjunction).

3.4 Section 5.5 Provisioning Sequence

In section 5.5 figures 7 and 8 have been updated to remove the ambiguity around the term "In service". This could have been assumed to mean the dataplane is activated. However the intent (with the decoupling of the PSM and dataplane activation) is to indicate that the "In service" is a primer for the dataplane to be activated (if it wasn't already so).

3.5 Section 6.3.2 Message checks.

Figure 12 has been added to section 6.3.2 to show an Example of SOAP fault translation to NSI failed message. The following explanatory text has also been added:

If any of the above parameters are malformed or omitted from the request message, the provider may not have the necessary information to return a failed or error message using the (asynchronous) callback mechanism. As such, the provider can use a (synchronous) SOAP fault to indicate a problem. If the requester receiving the SOAP fault is an AG, it should not propagate the SOAP fault up stream verbatim, but translate it into an appropriate failed or error message. The example (see Figure 12) below shows how a SOAP fault generated due to a malformed *reserve* message is translated by the AG to a *reservedFailed* message to the uPA.

3.6 Section 7.1.3: Correlation Ids and Failure Recovery

The aggregator has been amended to include an additional explanation of how the *reserveFailed* message from child NSAs are to be aggregated before passing up the tree. Change the coordinator functionality to get all *failed* messages back to the requester instead of just the first one as defined in the pseudo code in version 2.0.

3.7 Section 7.1.5 Per reservation information elements.

The following bullet has been added to this section:

If an RA receives a Connection request with a *startTime* in the past, this should be treated as 'now'. The RA should not change the *startTime* and keep it as part of the record of the reservation.

3.8 Section 7.1.6 Reservation Versioning Information

To support the modification of reservations, the notion of versioning has been introduced to identify the instance of a reservation over its lifetime.

- Versioning MUST be used as follows:
- Version numbers are integer values ≥ 0 (zero)
- Version numbers are assigned by the RA when a reservation request (i.e. NSI_rsv.rq) is made to a PA
- If a version number is not specified in an NSI_rsv.rq, it is assumed to be 0 (zero) regardless of whether the request is the initial or a subsequent request.
- An NSI_rsv.rq with a version number ≤ the (highest) current committed reservation version number will result in a failed request and an appropriate error

3.9 Section 9.3.3 added feedback parameter

The following row has been added to table 11:

Feedback	O	Provides a mechanism for a *ServiceException* to provide type specific feedback corresponding to this variable.

3.10 Section 9.4.10 text removed

The following text has been removed from Section 9.4.10 *dataPlaneStateChange* message elements:

> The originating *connectionId* and uPA are provided in separate elements to maintain the original context generating the data plane state change. The *timeStamp* is populated by the originating PA and propagated up the tree untouched by intermediate NSA.

This text problem was caused by the author copying the text from section 9.4.9 (*errorEvent*) into section 9.4.10 then modifying. The quoted text has been removed as it is not applicable to the *dataPlaneStateChange*.

3.11 Section 9.4.12.1: Query* operations.

There is a limitation in CS v2.0 in the definition of all of the *query* operations that is removed in v2.1.

The explicit access control rule is "reservation instances between the RA-PA pair" and appears in all query* operation definitions. This errata removes this restriction, allowing an RA to access all reservations within a PA given local policy definitions. Each PA is then free to enforce access control policies on this operation similar to other operations. It is important that we do not hinder the ability for a PA to support administration level users whom have access to all reservations.

The v2.0 text reads:

> 8.4.12.1 Request: *querySummary*
> The *querySummary* message provides a mechanism for an RA to query the PA for a set of connection service reservation instances **between the RA-PA pair**. This message can be used to monitor the progress of a reservation.

In v2.1 this has been replaced with:

> 9.4.12.1 Request: *querySummary*
> The *querySummary* message provides a mechanism for an RA to query the PA for a set of connection service reservation instances. This message can be used to monitor the progress of a reservation.

The following text has also been added:

> The PA may restrict visibility to reservations based on local access control policies.

The text has been similarly updated in the following sections:
9.4.12
9.4.13
9.4.14
9.5.1.27

3.12 Section 9.4.12.1: Request: querySummary

The change to allow an RA to retrieve all reservations within an RA pair results in a potentially large response to the query. To manage the amount of information returned the *ifModifiedSince* element has been added to section 9.4.12.1. This is described with the following new text:

> The *ifModifiedSince* element was added to the *querySummary* message with the goal of reducing reduce the amount of reservation information returned in query retrieval. It accomplishes this by allowing the user to specify a *dateTime* value in the request as returned from the last query. The target NSA uses this *dateTime* context to exclude reservations that have not changed since the last query.
>
> If an NSA receives a *querySummary* message containing an *ifModifiedSince* element, then it only returns those reservations matching the filter elements (connectionId, globalReservationId) if the reservation has been created, modified, or has undergone a change since the specified *ifModifiedSince* time. This includes user-initiated actions such

as provision and release, as well as state changes caused by events such as *dataPlaneStateChange* notifications (in dataPlaneStatus).

Updated Figure 65 to include the *ifModifiedSince* member element:

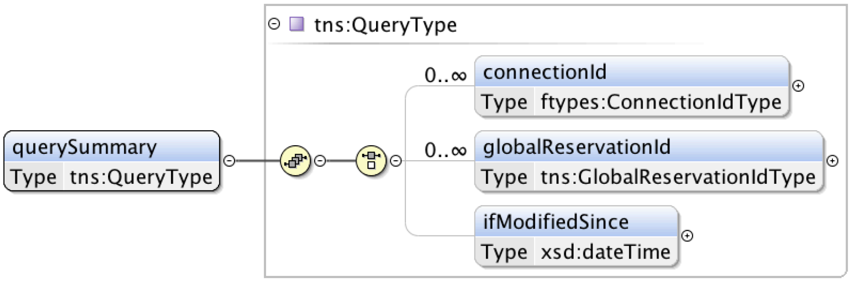

Figure 65 – *querySummary* request message structure.

The *ifModifiedSince* parameter has been added to table 45:

ifModifiedSince	If an NSA receives a querySummary message containing this element, then the NSA will only returns those reservations matching the filter elements (connectionId, globalReservationId) if the reservation has been created, modified, or has undergone a change since the specified ifModifiedSince time. This includes user-initiated actions such as provision and release, as well as state changes caused by events such as dataPlaneStateChange notifications (in dataPlaneStatus).

3.13 Section 9.4.12.2: Confirmation: *querySummaryConfirmed*
The following text has been added:

> In the *lastModified* element of this response the provider NSA includes the update time of the most recently created/modified/updated reservation on the system. The *lastModified* element is included even if the request did not include an *ifModifiedSince* element, and if the response does not contain any reservation results. This *lastModified* value can be used in the next query for this filter. The *lastModified* element will only be absent if the NSA does not support the *ifModifiedSince* capability.

Updated Figure 66 to include the *lastModified* member element:

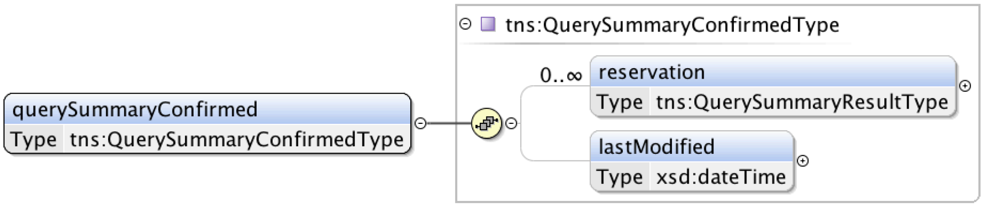
tns:QuerySummaryConfirmedType
querySummaryConfirmed
Type tns:QuerySummaryConfirmedType
0..∞
reservation
Type tns:QuerySummaryResultType
lastModified
Type xsd:dateTime

Figure 67 – *querySummaryConfirmed* message structure.

The following parameter has also be added to table 46:

lastModified	Includes the update time of the most recently created/modified/updated reservation on the system. The *lastModified* element is included even if the request did not include an *ifModifiedSince* element, and if the response does not contain any reservation results. This *lastModified* value can be used in the next query for this filter. The lastModified element will only be absent if the NSA does not support the ifModifiedSince capability.

3.14 Section 9.4.13: querySummarySync message elements

This section has the same change as documented for section 9.4.12.1

3.15 Section 9.5.1.16 QueryFailedType removed

The *QueryFailedType* type is not relevant, this was left in by mistake in v2.0. The following section has been removed: NSI CS v2.0 section 8.5.1.16 QueryFailedType does not appear as section 9.5.1.16 in v2.1.

3.16 Section 9.5.1.24 *QuerySummaryConfirmedType*

The following parameter has been added to table 89:

lastModified	O	Includes the update time of the most recently created/modified/updated reservation on the system. The *lastModified* element is included even if the request did not include an *ifModifiedSince* element, and if the response does not contain any reservation results. This *lastModified* value can be used in the next query for this filter. The *lastModified* element will only be absent if the NSA does not support the *ifModifiedSince* capability.

3.17 Section 9.5.1.27

This section has the same change as documented for section 9.4.12.1

3.18 Section 9.5.1.27 *QueryType*

The following text has been added to table 92:

ifModifiedSince	O	If an NSA receives a querySummary or querySummarySync message containing this element, then the NSA will only returns those reservations matching the filter elements (connectionId, globalReservationId) if the reservation has been created, modified, or has undergone a change since the specified ifModifiedSince time. This includes user-initiated actions such as provision and release, as well as state changes caused by events such as dataPlaneStateChange notifications (in dataPlaneStatus).

3.19 Section 9.5.1.30: updated versioning number to start with 1

Sections 7.1.6 and 9.5.1.30 were in conflict in v2.0. Section 9.5.1.30 was written when we were using "0" as a special version number to show the currently uncommitted first reservation criteria. This decision was later changed as described in 7.1.6. The text in 9.5.1.30 has been updated in v2.1 to reflect this:

> The version value specified in a reservation or modify request MUST follow versioning rules as defined in section 7.1.6.

3.20 Section 9.5.1.35: *ScheduleType*

StartTime and *endTime* definitions have been updated to allow for nil values to be entered.

The following text has been added to section 9.5.1.35:

> ***Parameters***
>
> The *ScheduleType* has the following parameters (M = Mandatory, O = Optional):

Parameter	M/O	Description
startTime	O nillable	The start time of the reservation.

		A start time of "now" is represented by a nil value in the startTime element. For backwards compatibility an absent startTime element in the inital reserve message also represents a start time of "now". An absent startTime element in a modification operation indicates there is no change to startTime. A startTime element with a nil value within a modify request represents a modification of startTime to "now". If a reserve request has a startTime in the past it should be considered as a start time of "now".
endTime	O nillable	The end time of the reservation. An "indefinite" end time is represented by a nil value in the endTime element. For backwards compatibility an absent endTime element in the inital reserve message also represents an "indefinite" end time. An absent endTime element in a modification operation indicates there is no change to endTime. An endTime element with a nil value within a modify request represents a modification of endTime to "indefinite". If a reserve request has a endTime in the past it should be considered as an invalid reservation request.

Table 1 *ScheduleType* message parameters

The following schedule element shows an example where both a startTime and endTime have been specified.

```
<schedule>
   <startTime>2016-03-29T14:09:00.000-07:00</startTime>
   <endTime>2016-03-29T14:24:00.000-07:00</endTime>
</schedule>
```

This schedule element shows an example of a “nil” startTime element indicatating a reservation start time of “now”.

```
<schedule>
   <startTime xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
   <endTime>2016-03-29T14:24:00.000-07:00</endTime>
</schedule>
```

For backwards compatibility in the initial reservation request, the following should be considered equivalent to the previous start time of “now” example:

```
<schedule>
   <endTime>2016-03-29T14:24:00.000-07:00</endTime>
</schedule>
```

This *schedule* element shows an example of a “*nil*” *endTime* element indicating an “indefinite” reservation end time.

```
<schedule>
   <startTime>2016-03-29T14:09:00.000-07:00</startTime>
   <endTime xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
</schedule>
```

For backwards compatibility in the initial reservation request, the following should be considered equivalent to the previous “indefinite” end time example:

```
<schedule>
  <startTime>2016-03-29T14:09:00.000-07:00</startTime>
</schedule>
```

3.21 Changes to Appendix B: Error Messages and Best Practices

The error message details have been moved and now belong in a separate error codes document. The reasoning behind this is that we expect the error codes to change based on implementation experience. The new document will allow error codes to be updated easily without needing to re-issue the CS document.

3.22 Changes to Appendix D: Formal Statement of Coordinator

The pseudo code in this appendix has been updated in v2.1 to show how the *reserveFailed* message from child NSAs are aggregated before passing up the tree. The coordinator functionality has been changed to get all *failed* messages back to the requester instead of just the first one as defined in the pseudo code shown in v2.0.

3.23 Appendix E: Service Definition Schema, Section 15.3

In section 15.3, table 105 – ‘NSI-CS point-to-point service-specific errors’ has been removed as error codes now belong in a separate document.

3.24 Appendix E: Service Definition Schema, Section 15.5.1.3 *TypeValueType*

A type definition for a type and value tuple used to represent simple parameter values has been added to the schema.

3.25 Appendix E: Service Definition Schema, Section 15.5.1.4 *clusionType*

A type definition for a *clusionType* has been added to the schema. This type definition is used to model pathfinding inclusions/exclusions in a point-to-point service request. The possible types and values specified in an inclusion or exclusion will be defined in the service specific *Service Definition*.

The following parameters are added:

Parameters

The *ClusionType* has the following parameters (M = Mandatory, O = Optional):

Parameter	M/O	Description
type	M	Order attribute is provided only when the STP is part of an *orderedSTP*List.
lt	O	The Service Termination Point (STP).
lte	O	Less than equal to conditional element.
gt	O	Greater than conditional element.
gte	O	Greater than equal to conditional eement.
eq	O	Equal conditional element.

In addition some examples of how to use the *clusionType* are now included in this section.

3.26 Appendix E: Service Definition Schema, Section 15.6

In Section 15.6 Reservation request, the example *reserve* request XML message for a bidirectional service has been updated.

3.27 Appendix E: Service Definition Schema, Section 15.7

In Section 15.7 Reservation modification, the example modification using a *reserve* request XML message for a bidirectional service has been updated.

3.28 Old Appendix F

Appendix F that was in v2.0 has been removed from v2.1 as this subject it is covered in more detail in the pathfinding and signaling document.

3.29 New Appendix F: Using the Explicit Routing Object in practice

A new appendix F has been added to bring in relevant text from the old document draft-gwdi-macauley-nsi_ero-v03 to the CS document. Note that the old ero document is now obsoleted by this appendix.

4. Contributors

Guy Roberts, GÉANT
John MacAuley, ESnet
Chin Guok, ESnet

5. 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.

6. 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.

7. Full Copyright Notice