# CS v2.0 edits

2 May 2014, Guy Roberts

**\* C.S. 2.0 documentation remarks.**The table numbers are wrong. Appendix A starts with table numbers from 66,
while the last table in the main part is 129. No duplicate table numbers
please. This is for draft 25.

Fixed

**\* Provision/Release/Modify**How do deal with limited NRM/NSA functionality here?
Are the operations mandatory and dummy/noops, or does one reject not
supported requests with a NOT\_IMPLEMENTED error. The expected behaviour
should be described so that incompatibilities between NSAs can be avoided.

Guy proposal:

The release/provision/modify functionality is optionally supported in a PA. To ensure correct transitions of the statemachine, all transitions MUST be carried out as defined in the NSI statemachines regardless of whether the release/provision actions are actually performed.

* **Release**: If a PA does not support the provision/release cycle on an existing reservation, then the PA MUST spoof a *releaseConfirm* in response to a *release* request. i.e. a response is returned even though there has been no data-plane affecting changes.
* **Provision**: PA MUST operate the first *provision* correctly. If a PA does not support the provision/release cycle on an existing reservation, then the PA MUST spoof a *provisionConfirm* in response to a *provision* request. I.e a response is returned even though there has been no data-plane affecting changes.
* **Modify**. If the modify functionality is not supported by a PA, then a *reservedFailed* message MUST be returned with a ‘not implemented’ error when an attempt is made to modify an existing reservation. When an RA receives a ‘not implemented’ error, this is a considered a reserve fail event. When the Agg receives a ‘not implemented’ error, this is forwarded up the tree.

**\* Start & End Time**Does start/end time specify when the link is available or when resource are allocated, and hence when the provisioning can start. This problem is also known as the `guard band' problem. There are two different guard bands, a circuit provisioning guard band and a circuit deprovsioning guard band.

The current documentation describes that all guard bands start from the moment a request has been received (including the automatic tear down at end time). This works without any problem if all guard band fall between start and end time, but can cause problems when the deprovisioning guard band will only end after end time of the reservation (creating a period of resource unavailability while the administration says that the resources should be available). There are implementations that force the guard band to always be between start and end time and will start deprovisioning sometime before the reservation is scheduled to end (which may cause unexpected state change messages).

The following clarification text is proposed:

Section 4.5: Guardbands

There may be a delay between the requested in-service start time and the activation of the data plane. So that the RA knows that the data-plane has actually changed state, a state change notification is defined. The *dataPlaneStateChange* notification is sent to the RA that issued the original *reserve* request when the data-plane status has changed. Possible data-plane status changes are: activation, deactivation and activation version change.

**Start Time.** The start time is the earliest that the activation can occur. There may be a delay in completing the activation depending on the time taken by the NRM to perform the activation.

In the situation where the RA wishes to ensure that the activation has completed at a guaranteed point in time, it is the responsibility of the RA to add a guard band as they see fit to the start time. The RA is responsible for choosing an appropriate guard time based on their knowledge of the expected provisioning delay at the target NRM.

**End Time.** The end time is the earliest that the deactivation can occur. There may be a delay in completing this action depending on the time taken by the NRM to complete the deactivation.

In the situation where the RA wishes to ensure that the deactivation has either started before or completed after at a guaranteed point in time, it is the responsibility of the RA to add a guard band as they see fit to the end time. The RA is responsible for choosing an appropriate guard time based on their knowledge of the expected deactivation delay at the target NRM.

**\* querySummary**Is the criteria mandatory in the payload?
As far as we know this was not mandatory in CS 1.0, but it is unclear if it is or is not mandatory in CS 2.0.

Paste this line into the relevant schema section to clarify the correct behaviour:

“A query will return the currently committed reservation version number, however, if the initial version of the reservation has not yet been committed, the query will return base reservation information (*connectionId*, *globalReservationId*, *description*, *requesterNSA*, and *connectionStates*) with no versioned reservation criteria.”

**\* Security Attributes**Why are they called session security attributes, when we do not have a session concept in NSI?
Basically everything is allowed as AttributeValue. Some clear restrictions / patterns are needed if we want this to work.
The schema is overly generic and allows for multiple representations of the same thing. I.e., attributes on sessionSecurityAttributes, multiple Attributes with same Name. Further the anyType under AttributeValue makes parsing and passing a complete pain. Essentially we have a key that maps to a list of values, but a BCP for the schema is clearly needed. I (HTJ) expect these things to be made clear in the upcoming AAI document.

Though they may be misleading, there will be no change to the ‘sessionSecurityAttributes’ naming as we don’t want to issue a new schema for such a minor issue.

The following clarification is to be added to the CS document: “An NSA AG MUST pass any received session security attributes on to all children. “

The other issues raised by Henrik will be picked up in the AAI practices document.

**\* Label - STP Mapping**In STPs a label is denoted as "vlan"
In NML a label is denoted under a namespace, e.g., “<http://schemas.ogf.org/nml/2012/10/ethernet#vlan>"
Where is this mapping defined? Is it a 1:1 mapping? (I hope so).

I note that the NSI topology doc describes the label and how this maps to a vlan:

"Endpoints in a network often have a technology label associated with them, for example VLANs or wavelengths. Rather than describing each of these available labels as individual STPs, we introduce the STP Group, equivalent to an NML PortGroup.  An STP with a speciﬁc label can then be selected using the query component syntax as speciﬁed in [? ], so for example: urn:ogf:network:example.net:2013:A2?vlan=1781 is a way to phrase a request to an STP with VLAN 1781 part of the STP Group identiﬁed by urn:ogf:network:example.net:2013:A2."

By inference the urn:ogf:network:example.net:2013:A2?vlan=1781 has a direct 1:1 mapping to the NML namespace. Do we need to spell this out?

An update may be needed in NSI topology doc, however no changes will be made to the CS document.

**\* STP URN Conformance**According to RFC2141 the character "?" is reserved for future usage and not allowed in URNs.
I don't think we should change to current structure, but just not call it a urn (despite it starting with the urn prefix).
Not relevant to the CS document. Update may be needed in that document

In the CS document, all references to the ‘?’ in URNs will be removed. This issue will be addressed in updates to the NSI Topology document.

**\* queryRecursive**What is the semantics when querying with global reservation id, and getting multiple back (clash avoidance is not possible). This allows for some very wonky stuff.
My (HTJ) suggested solution it so disallow global reservation id in the queryRecursive primitive. I have done this in OpenNSA. Similarly I do not allow blank queries for queryRecursive.

If queryRecursive fails down the tree, should a partial tree(s) be returned or a failure returned? An interesting case is when multiple connections with different tree are queried. Here the information for one connection might be complete, but the tree for another connection is cannot be completed. Returning failure or partial results are both
sub-optimal solutions.

Similarly, OpenNSA currently only allows a single connection id in queryRecursive to avoid this situation (and to discourage full-tree queries).

NSI requires that we trust client to create a globally unique reservation Id. However the responsibility for ensuring that globalIds are truly globally unique is outside the scope of this document. This is the responsibility of the entity or organization that issues the globalIds.