

NSIS  
Internet-Draft  
Expires: August 25, 2005

X. Fu  
Univ. Goettingen  
H. Tschofenig  
T. Tsenov  
Siemens  
February 21, 2005

QoS NSLP State Machine  
draft-fu-nsis-qos-nslp-statemachine-01.txt

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of [Section 3 of RFC 3667](#). By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with [RFC 3668](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 25, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document describes the state machines for the NSIS Signaling Layer Protocol for Quality-of-Service signaling (QoS NSLP). A set of state machines for QoS NSLP entities at different locations of a flow path are presented in order to illustrate how QoS NSLP may be

Internet-Draft

QoS NSLP State Machine

February 2005

implemented.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Notational conventions used in state diagrams . . . . .	<a href="#">5</a>
<a href="#">4.</a>	State Machine Symbols . . . . .	<a href="#">7</a>
<a href="#">5.</a>	Common Rules . . . . .	<a href="#">8</a>
<a href="#">5.1</a>	Common Procedures . . . . .	<a href="#">8</a>
<a href="#">5.2</a>	Common Variables . . . . .	<a href="#">8</a>
<a href="#">5.3</a>	Constants . . . . .	<a href="#">9</a>
<a href="#">5.4</a>	Assumptions . . . . .	<a href="#">9</a>
<a href="#">6.</a>	State machine for QNI QoS NSLP node . . . . .	<a href="#">10</a>
<a href="#">7.</a>	State machine for QNE QoS NSLP nodes . . . . .	<a href="#">13</a>
<a href="#">8.</a>	State machine for QNR QoS NSLP node . . . . .	<a href="#">17</a>
<a href="#">9.</a>	Security Considerations . . . . .	<a href="#">20</a>
<a href="#">10.</a>	Open Issues . . . . .	<a href="#">21</a>
<a href="#">11.</a>	Change History . . . . .	<a href="#">22</a>
<a href="#">11.1</a>	Changes in Version -01 . . . . .	<a href="#">22</a>
<a href="#">12.</a>	Acknowledgments . . . . .	<a href="#">23</a>
<a href="#">13.</a>	References . . . . .	<a href="#">24</a>
<a href="#">13.1</a>	Normative References . . . . .	<a href="#">24</a>
<a href="#">13.2</a>	Informative References . . . . .	<a href="#">24</a>
	Authors' Addresses . . . . .	<a href="#">24</a>
	Intellectual Property and Copyright Statements . . . . .	<a href="#">26</a>

## [1.](#) Introduction

This document describes the state machines for QoS NSLP [[1](#)], trying to show how QoS NSLP can be implemented to support its deployment. The state machines described in this document are illustrative of how the QoS NSLP protocol defined in [[1](#)] may be implemented for the QNI QoS NSLP node, QNE QoS NSLP nodes, and QNR QoS NSLP node in the flow path. Where there are differences [[1](#)] are authoritative. The state machines are informative only. Implementations may achieve the same results using different methods.

According to [[1](#)], there are several possibilities for QoS NSLP signaling, at least including the following:

- end-to-end signaling vs. scoped signaling
- sender-initiated signaling vs. receiver-initiated signaling

(which need to be incorporated into use scenarios when describing state machine. Note they are represented by way of certain objects/flags in Reserve and Query messages.)

The messages used in the QoS NSLP protocol can be summarized as follows:

Requesting message	Responding message
RESERVE	None or RESERVE or RESPONSE
QUERY	RESERVE or RESPONSE
RESPONSE	NONE
NOTIFY	NONE

We describe a set of state machines for different roles of entities running QoS NSLP to illustrate how QoS NSLP may be implemented.

## [2.](#) Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[2\]](#).

### [3.](#) Notational conventions used in state diagrams

The following text is reused from [\[3\]](#) and the state diagrams are based on the conventions specified in [\[4\]](#), Section 8.2.1. Additional state machine details are taken from [\[5\]](#).

The complete text is reproduced here:

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e.,

a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow; i.e., no specific state is identified as the origin of the transition. When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions, and once asserted remains asserted until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic; i.e., execution of a procedure completes before the next sequential procedure starts to execute. No procedures execute outside of a state block. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied, and all procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable

that is set to a particular value in a state block retains this value until a subsequent state block executes a procedure that modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary.

In addition to the above notation, there are a couple of clarifications specific to this document. First, all boolean variables are initialized to FALSE before the state machine execution begins. Second, the following notational shorthand is specific to this document:

`<variable> = <expression1> | <expression2> | ...`

Execution of a statement of this form will result in `<variable>` having a value of exactly one of the expressions. The logic for which of those expressions gets executed is outside of the state machine and could be environmental, configurable, or based on another state machine such as that of the method.

#### [4.](#) State Machine Symbols

- ( ) Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.
- ; Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal language conventions for reading text.

= Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., `a = b = X` the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.

! Logical NOT operator.

&& Logical AND operator.

|| Logical OR operator.

if...then... Conditional action. If the Boolean expression following the if evaluates to TRUE, then the action following the then is executed.

\{ statement 1, ... statement N \} Compound statement. Braces are used to group statements that are executed together as if they were a single statement.

!= Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.

== Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.

> Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.

<= Less than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either less than or equal to the value of the expression to the right.

++ Increment the preceding integer operator by 1.



Throughout the document we use terms defined in the [1], such as flow sender, flow receiver, QUERY, RESERVE or RESPONSE.

## 5.1 Common Procedures

tx\_RESERVE(Toff): Transmit RESERVE message with 'Teardown' bit off  
tx\_RESERVE(Ton): Transmit RESERVE message with 'Teardown' bit on  
tx\_RESPONSE(): Transmit RESPONSE message  
tx\_QUERY(<object>): Transmit QUERY message with <object>  
tx\_QUERY(w/o<object>): Transmit QUERY message without <object>  
tx\_NOTIFY(): Transmit NOTIFY message  
rx\_RESPONSE(): Receive RESPONSE message  
rx\_QUERY(): Receive QUERY message  
rx\_RESERVE(): Receive RESERVE message  
rx\_NOTIFY(): Transmit NOTIFY message  
TIMEOUT\_StateLifetime: State lifetime timer expiration  
TIMEOUT\_Refresh: Refresh interval timer expiration  
TIMEOUT\_Refresh: Wait-Response interval timer expiration  
tg\_QUERY: External trigger to send a QUERY message (typically triggered by the application).  
tg\_RESERVE: External trigger to send a RESERVE message.  
tg\_TEARDOWN: External trigger to clear previously established QoS state (typically triggered by the application). It is translated to a tx\_RESERVE(Ton) message.  
Install QoS state: Install the local QoS state.  
Refresh QoS state: Refresh the local QoS state.  
Delete QoS state: Delete the local QoS state.  
Send info to Application: Report information to the application.  
RMF: Performs Resource Management Function and returns the following values{AVAIL, NO\_AVAIL}.  
SetRII: Sets the RII object of the messages e.g. the node requests explicit response to the message being sent. Returns values {0,1}.  
CheckRII: Checks the RII object of received RESPONSE message if it is requested by current node or other upstream node. Returns values {LOCAL, NO\_LOCAL}.  
ProcessQUERY: Processes a Query message and provides the requested info

## 5.2 Common Variables

RII: Request Identification Information (RII) object. Logical variable representing if the RII is set or not. Takes values {0,1}.

SCOPING: Scoping flag of common message header. Takes values {"Next\_hop", "Whole\_path"}.

RSN: Reservation Sequence Number object. Takes values:

- recRSN - RSN object of the received message
- currRSN - Current stored RSN value for installed QoS state. (Assumed to be the one for the direction where the message comes from e.g.Upstream/Downstream)

ACK: Acknowledgement flag of common message header. Takes values {"On", "Off"}.

SummaryRefresh: Keeps information if Summary refresh method may be used for refreshing a installed QoS state. Takes value {"On", "Off"}.

E\_SPEC: Error\_Spec object. Takes values:

- 0x02? - Success values
- 0x04? - Transient Failure values
- ERROR - Not specified in the QoS NSLP draft, but used here. ([Section 10](#))

QSPEC: QoS specification object.

FlowID: Flow ID kept by the installed QoS state.

Replace: Replace flag of common message header. Takes values {"On", "Off"}.

SII: Source Identification Information entry. Takes values:

- CurrSII - SII entry stored for current installed QoS state. (Assumed to be the one for the direction where the message comes from e.g.Upstream/Downstream)
- newSII - SII of the received message is different from the SII stored for the current installed QoS state.

### [5.3](#) Constants

### [5.4](#) Assumptions

- o For simplification not all included objects in a message are showed. Only those that are significant for the case are showed. State machines do not present handling of messages that are not significant for management of the states such as certain NOTIFY and QUERY messages.
- o State machines represent handling of messages of the same Session ID and with no protocol errors. Separate parallel instances of the state machines should handle messages for different Session IDs.
- o Default message handling should be defined for messages with different Session IDs that have impact on current session state and error messages. This is not included in the current version.
- o ACK flag in the common header is set "On" by default.

- o Direction of receiving and sending messages is not specified. We assume it is implicit from the context.

## 6. State machine for QNI QoS NSLP node

```
-----
State: INIT
-----
```

Condition	Action	State
UCT	initialize variables	IDLE

```
-----
State: IDLE
-----
```

Condition	Action	State	Note
rx_QUERY(RII)	ProcessQUERY   tx_RESPONSE(RII)	IDLE	
(tg_RESERVE) && (RMF="NO_AVAIL")	Send info to Application	IDLE	
(tg_RESERVE) && (!RII) && (!setRII) && (RMF="AVAIL")	tx_RESERVE(w/oRII),   Install QoS state,   Send info to Application	QoS state   Installed	
(tg_RESERVE) && (setRII) &&(RMF="AVAIL")	Install QoS state,   tx_RESERVE(RII)	QoS state   Installed +   WAITRESP2	
(rx_QUERY)&&(!RII)&& (RMF="NO_AVAIL")	Tx_RESPONSE(RSN,   E_SPEC="ERROR")	IDLE	1)   2)
(rx_QUERY) && (!RII) && (!setRII) && (RMF="AVAIL")	tx_RESERVE(w/oRII),   Install QoS state,   Send info to Application	QoS state   Installed	2)

(rx_QUERY) && (!RII) && (setRII) && (RMF="AVAIL")	Install QoS state, tx_RESERVE( RII)	QoS state Installed + WAITRESP2	2)
(tg_QUERY) && (setRII)	tx_QUERY(RII)	WAITRESP1	
-----	-----	-----	-----

## Note:

- 1) How to signal unsuccessful reservation for Receiver initiated reservation (No RII included; sent Response(RSN) should not be forwarded further than the next peer). No Error\_SPEC value specified for this case.
- 2) Relevant for Receiver-initiated reservation.

-----  
State: WAITRESP1  
-----

Condition	Action	State
(TIMEOUT_WaitResp) && (!MaxRetry)	tx_QUERY(RII)	WAITRESP1
(TIMEOUT_WaitResp) && (MaxRetry)	Send info to Application	IDLE
rx_RESPONSE	Send info to Application	IDLE
-----	-----	-----

-----  
State: QoS state installed + WAITRESP2  
-----

Condition	Action	State
(TIMEOUT_WaitResp) &&	tx_RESERVE(RII)	QoS state

(!MaxRetry)		installed + WAITRESP2
(TIMEOUT_WaitResp) && (MaxRetry)	Delete QoS state Send info to Application	IDLE
rx_RESPONSE(RII, E_SPEC="0x04?")	Delete QoS state Send info to Application	IDLE
rx_RESPONSE(RII, E_SPEC="0x02?")	Send info to Application SummaryRefresh="On"	QoS state installed

-----  
State: QoS state installed  
-----

Condition	Action	State	Note
TIMEOUT_Refresh	If (SummaryRefresh="On") (Tx_RESERVE(RSN)) && (SummaryRefresh="Off") Else Tx_RESERVE(RSN,QSPEC);	QoS state installed	
rx_RESPONSE(RSN, E_SPEC="0x02?")	SummaryRefresh="On"	QoS state installed	
TIMEOUT_StateLifetime	Delete QoS state Send info to Application	IDLE	1)
tg_TEARDOWN	Delete QoS state, tx_RESERVE(Ton)	IDLE	
rx_NOTIFY(RSN, E_SPEC="0x04?")	Delete QoS state Send info to Application	IDLE	

Note:

- 1) If QoS state lifetime expires in QNI, should RESERVE(Ton) be sent downstream the path?

Fu, et al.

Expires August 25, 2005

[Page 12]

Internet-Draft

QoS NSLP State Machine

February 2005

## [7.](#) State machine for QNE QoS NSLP nodes

-----  
State: INIT  
-----

Condition	Action	State
UCT	initialize variables	IDLE

-----  
State: IDLE  
-----

Condition	Action	State	Note
-----------	--------	-------	------

(rx_QUERY) && (!RII)	tx_QUERY(w/oRII)	IDLE	2)
(rx_QUERY(RII, SCOPING="Next_hop")	ProcessQUERY, Tx_RESPONSE(RII)	IDLE	7)
(rx_QUERY) && (RII)	tx_QUERY(w/RII)	IDLE	7)
(rx_RESERVE(RII)) && (RMF="NO_AVAIL")	Tx_RESPONSE(RII, E_SPEC="0x04?")	IDLE	3)
(rx_RESERVE) && (!RII)&& (RMF="NO_AVAIL")	Tx_RESPONSE(RSN, E_SPEC="0x04?")	IDLE	3)
(rx_RESPONSE(RII)) && (CheckRII="Not_LOCAL")	Tx_RESPONSE(RII)	IDLE	
(rx_RESERVE)&& !(setRII) && (RMF="AVAIL")	Install QoS state, If(ACK="On") Tx_RESPONSE(RSN, E_SPEC="0x02?"); If(RII) Tx_RESPONSE(RII) Else Tx_RESPONSE(w/oRII)	QoS State Installed	1a)
(rx_RESERVE(SCOPING= "Next_hop")) && (RMF="AVAIL")	Install QoS state, If(RII) Tx_RESPONSE(RII, E_SPEC="0x02?") Else Tx_RESPONSE(RSN, E_SPEC="0x02?")	QoS State Installed	1b)

(rx_RESERVE) && (setRII) && (RMF="AVAIL")	Install QoS state, Tx_RESPONSE(RII), If(ACK="On") Tx_RESPONSE(RSN, E_SPEC="0x02?");	QoS State Installed + WAITRESP1	4)
(tg_QUERY) && (setRII)	tx_QUERY(RII)	WAITRESP2	5)

State: QoS State Installed + WAITRESP1

-----

Condition	Action	State	Note
(TIMEOUT_WaitResp) && (!MaxRetry)	tx_RESERVE(RII)	WAITRESP1	
(TIMEOUT_WaitResp) && (MaxRetry) &&	Delete QoS State, tx_NOTIFY(RSN, E_SPEC="0x04?") Send info to Application	IDLE	
(rx_RESPONSE(RII, E_SPEC="0x04?")) &&(CheckRII="LOCAL")	Delete QoS State, tx_NOTIFY(RSN, E_SPEC="0x04?") Send info to Application	IDLE	4)
(rx_RESPONSE(RII, E_SPEC="0x02?")) &&(CheckRII="LOCAL")	Send info to Application SummaryRefresh="On"	QoS State Installed	

-----

State: WAITRESP2

-----

Condition	Action	State	Note
(TIMEOUT_WaitResp) && (!MaxRetry)	tx_QUERY(RII)	WAITRESP2	
(TIMEOUT_WaitResp) && (MaxRetry)	Send info to Application	IDLE	
(rx_RESPONSE) &&	Send info to Application	IDLE	

(CheckRII="LOCAL")			
-----	+	-----	+



-----  
State: QoS State Installed  
-----

Condition	Action	State	Note
rx_RESERVE(Ton)	tx_RESERVE(Ton), Delete QoS state	IDLE	
rx_RESERVE	Refresh QoS state If(ACK="On") Tx_RESPONSE(RSN, E_SPEC="0x02?")	QoS State Installed	6)
rx_RESPONSE(RSN, E_SPEC="0x02?")	SummaryRefresh="On"	QoS State Installed	6)
TIMEOUT_Refresh	If (SummaryRefresh="On") (Tx_RESERVE(RSN)) &&(SummaryRefresh="Off") Else Tx_RESERVE(RSN,QSPEC)	QoS State Installed	6)
(rx_RESPONSE(RII, E_SPEC="0x02?")) &&(ChechRII="NOT_LOCAL")	SummaryRefresh="On" Tx_RESPONSE(RII, E_SPEC="0x02?")	QoS State Installed	
(TIMEOUT_StateLifetime)	Delete QoS state	IDLE	8)
(rx_RESPONSE(RII, E_SPEC="0x04?")) &&(ChechRII="NOT_LOCAL")	Delete QoS state rx_RESPONSE(RII, E_SPEC="0x04?")	IDLE	
rx_RESPONSE(RSN, E_SPEC="0x04?")	Delete QoS state rx_NOTIFY(RSN, E_SPEC="0x04?")	IDLE	
rx_NOTIFY(RSN, E_SPEC="0x04?")	Delete QoS state rx_NOTIFY(RSN, E_SPEC="0x04?")	IDLE	

(Rx_RESERVE)&&(currSII)	Update QoS state	QoS State	9)
&&(Replace="On")	If (RII)	Installed	
&&(RMF="AVAIL")	Tx_RESERVE(RII,QSPEC)		
&&((recRSN>=currRSN)	else		
(newFlowID))	Tx_RESERVE(RSN,QSPEC);		
	If (ACK="On")&&(!RII)		
	tx_RESPONSE(RSN,		
	E_SPEC="0x02?");		
(Rx_RESERVE)&&(newSII)	Update QoS state	QoS State	9)
&&(RMF="AVAIL")	If (RII)	Installed	
&&((recRSN>=currRSN)	Tx_RESERVE(RII,QSPEC)		
(newFlowID))	else		
	Tx_RESERVE(RSN,QSPEC);		
	If (ACK="On")&&(!RII)		
	tx_RESPONSE(RSN,		
	E_SPEC="0x02?");		
	If (Replace="On")		
	tx_Reserve(Ton)		
	to currSII		

## NOTE:

- 1) Successful reservation without Response request (1a) and with Scoping (1b).
- 2) Processing of Query msg for Receiver initiated reservation
- 3) Unsuccessful reservation with/without request for response from previous node in the path.
- 4) Unsuccessful reservation. RII requested at the local node. NOTIFY(RSN) is sent further to the upstream nodes.
- 5) Processing of Query msg triggered by the application layer.
- 6) QoS State refresh procedures
- 7) Processing of Query msg received from an upstream node.
- 8) We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers. These issues are not described in the QoS NSLP draft.
- 9) Update QoS state and Re-route functionality

Internet-Draft

QoS NSLP State Machine

February 2005

8. State machine for QNR QoS NSLP node

```

-----
State: INIT
-----

```

Condition	Action	State
UCT	initialize variables	IDLE

```

-----
State: IDLE
-----

```

Condition	Action	State	Note
rx_QUERY(RII)	tx_RESPONSE(RII)	IDLE	
(rx_RESERVE)&&(!RII) && (RMF="NO_A")	Tx_RESPONSE(RSN, E_SPEC="0x04?")	IDLE	
(rx_RESERVE(RII)) && (RMF="NO_A")	Tx_RESPONSE(RII, E_SPEC="0x04?")	IDLE	
(tg_QUERY) && (!setRII)	tx_QUERY(w/oRII)	WAITRESV	1)
(rx_RESERVE(RII)) && (RMF="AVAIL")	Install QoS state   Tx_RESPONSE(RII, E_SPEC="0x02?")	QoS state   installed	2)
(rx_RESERVE)&&(!RII) && (RMF="AVAIL")	Install QoS state   Tx_RESPONSE(RSN, E_SPEC="0x02?")	QoS state   installed	2)

Internet-Draft

QoS NSLP State Machine

February 2005

-----  
 State: WAITRESV  
 -----

Condition	Action	State	Note
TIMEOUT_WaitResp	Tx_QUERY(w/oRII)	WAITRESV	
(TIMEOUT_WaitResp) && (MaxRetry)	Send info to Appl.	IDLE	
(rx_RESERVE)&&(!RII) && (RMF="Not_Avail")	tx_RESPONSE(RSN, E_SPEC="0x04?") Send info to Appl.	IDLE	3)
(rx_RESERVE(RII)) && (RMF="Not_Avail")	tx_RESPONSE(RII, E_SPEC="0x04?") Send info to Appl.	IDLE	3)
rx_RESPONSE(E_SPEC= "ERROR")	Send info to Appl.	IDLE	4)
(rx_RESERVE)&&(!RII) && (RMF="AVAIL")	Install QoS state Tx_RESPONSE(RSN, E_SPEC="0x02?")	QoS state installed	
(rx_RESERVE(RII)) && (RMF="AVAIL")	Install QoS State Tx_RESPONSE(RII)	QoS state installed	

-----  
 State: QoS state installed  
 -----

Condition	Action	State	Note
rx_RESERVE	Refresh QoS state If(ACK="On") Tx_RESPONSE(RSN, E_SPEC="0x02?")	QoS state installed	
TIMEOUT_StateLifetime	Delete QoS state	IDLE	5)
rx_RESERVE(Ton)	Delete QoS state	IDLE	

Note:

- 1) Initiation of Receiver-side reservation
- 2) Successful Reservation with & without response request from the QNI side
- 3) Unsuccessful Reservation with & without response request from the QNI side.
- 4) How to signal unsuccessful reservation for Receiver initiated reservation(No RII is included, received Response(RSN) should

not be forwarded.

- 5) We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers. These issues are not described in the QoS NSLP draft.

## [9.](#) Security Considerations

This document does not raise new security considerations. Any security concerns with the QoS NSLP are likely reflected in security related NSIS work already (such as [\[1\]](#) or [\[6\]](#)).

For the time being, the state machines described in this document do not consider the security aspect of QoS NSLP protocol itself. A future versions of this document will add security relevant states and state transitions.

## [10](#). Open Issues

This document tries to describe possible states and transitions for QoS NSLP according to its current specification [[1](#)], Section 5. We found some issues during the development of the state machines.

- o For receiver-initiated reservation, it is unclear who triggers a teardown.
- o Bi-directional reservation is difficult to support as the state machine becomes quite complex (note at one particular point in time the protocol state engine can be only in one state).
- o How to signal unsuccessful reservation for Receiver initiated

reservation (No RII included; a resulting Response(RSN) cannot be forwarded further than the next peer). No Error\_SPEC value specified for this case.

- o If QoS state lifetime expires in QNI, should RESERVE(Ton) be sent downstream the path?
- o The case of unsuccessful reservation at a QNE node and no RII specified by upstream nodes. According to the spec RESPONSE(RSN) should not be forwarded further than the next peer. Currently we use NOTIFY(RSN) that is sent further to the upstream nodes.
- o We assume that handling of QoS state lifetime expiration event is based on the local policy of the node. NOTIFY/Reserve(Ton) messages might be sent to other peers. These issues are not described in the QoS NSLP draft.
- o The draft states that RESERVE message MUST be sent only towards the QNR. This is not the case when re-routing procedure is done and RESERVE(Ton) message should be sent from merging QNE node for deleting the old branch. We believe this is towards the QNI.
- o Re-routing functionality described in this document is not complete and need further consideration.

## [11.](#) Change History

### [11.1](#) Changes in Version -01

Version -01 covers more functionalities of the QoS NSLP. This



requires addition and changes of the notations. The main details are as follows:

1. Notation of the nodes changed to QNI, QNE and QNR.
2. Description of soft state refresh functionality.
3. Support of ACK flag in the common header.
4. Include of QoS NSLP objects, flags from the common header and entries stored with the installed QoS state in a node: ACK, Replace, RSN, Error\_SPEC, QSPEQ, FlowID, SII.
5. Initial description of Re-routing functionality ([Section 10](#)).
6. For support of all listed changes, some notations are changed.

## [12.](#) Acknowledgments

The authors would like to thank Sven Van den Bosch for his feedback.

Internet-Draft

QoS NSLP State Machine

February 2005

## 13. References

### 13.1 Normative References

- [1] Bosch, S., Karagiannis, G. and A. McDonald, "NSLP for Quality-of-Service signaling", Internet-Draft [draft-ietf-nsis-qos-nslp-05](#), October 2004.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

### 13.2 Informative References

- [3] Vollbrecht, J., Eronen, P., Petroni, N. and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", Internet-Draft [draft-ietf-eap-statemachine-06](#), December 2004.
- [4] Institute of Electrical and Electronics Engineers, "DRAFT Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control (Revision)", IEEE 802-1X-REV/D9, January 2004.
- [5] Ohba, Y., "State Machines for Protocol for Carrying Authentication for Network Access (PANA)", Internet-Draft [draft-ohba-pana-statemachine-01](#), February 2005.
- [6] Tschofenig, H. and D. Kroeselberg, "Security Threats for NSIS", Internet-Draft [draft-ietf-nsis-threats-06](#), October 2004.

## Authors' Addresses

Xiaoming Fu  
University of Goettingen  
Telematics Group  
Lotzestr. 16-18  
Goettingen 37083  
Germany

Email: fu@cs.uni-goettingen.de

Internet-Draft

QoS NSLP State Machine

February 2005

Hannes Tschofenig  
Siemens  
Otto-Hahn-Ring 6  
Munich, Bayern 81739  
Germany

Email: Hannes.Tschofenig@siemens.com

Tseno Tsenov  
Siemens  
Otto-Hahn-Ring 6  
Munich, Bayern 81739  
Germany

Email: tseno.tsenov@mytum.de

## Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat 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 IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Disclaimer of Validity

This document and the information contained herein are provided on an

"AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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 Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

#### Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.