# Network Services Interface

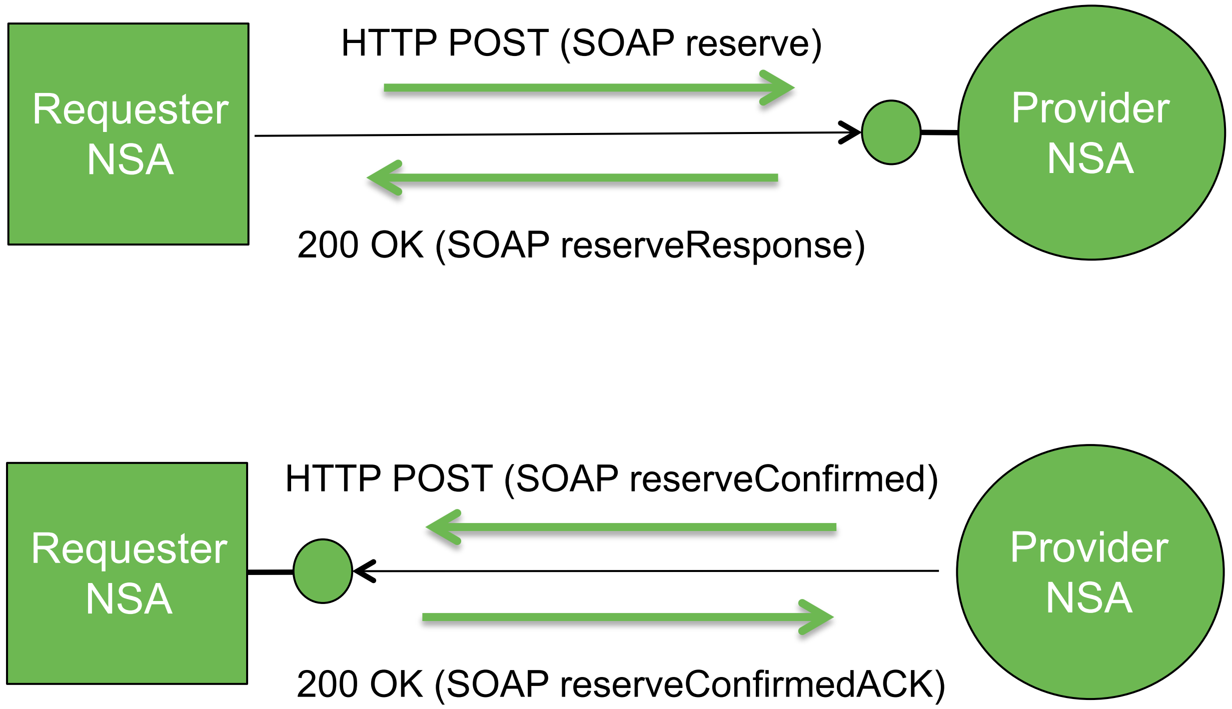
## OGF 38a, The Maastricht Sessions

Notification handling for polling clients
version 2

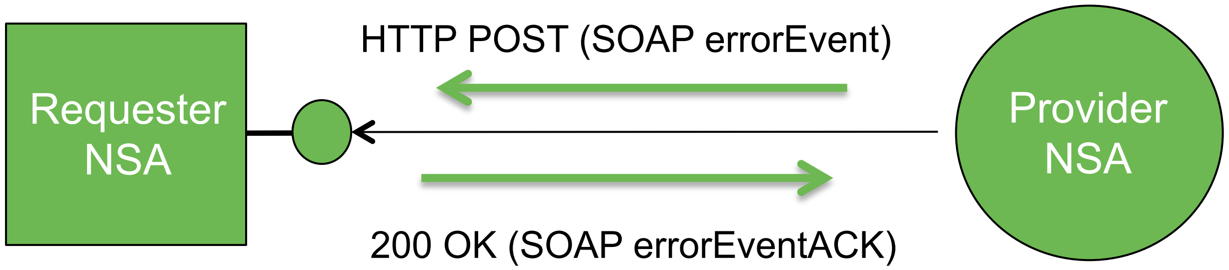
John MacAuley, SURFnet

13th November 2013

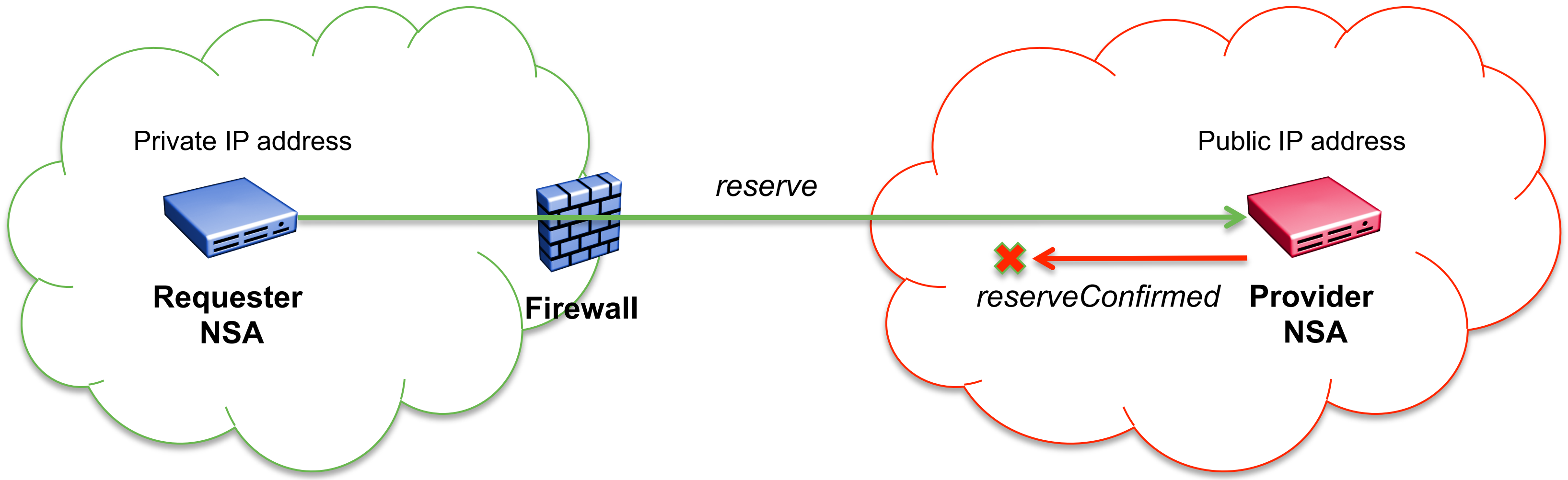
## Asynchronous Messaging


Requester NSA
HTTP POST (SOAP reserve)
Provider NSA
200 OK (SOAP reserveResponse)
Requester NSA
HTTP POST (SOAP reserveConfirmed)
Provider NSA
200 OK (SOAP reserveConfirmedACK)

# Asynchronous Notifications

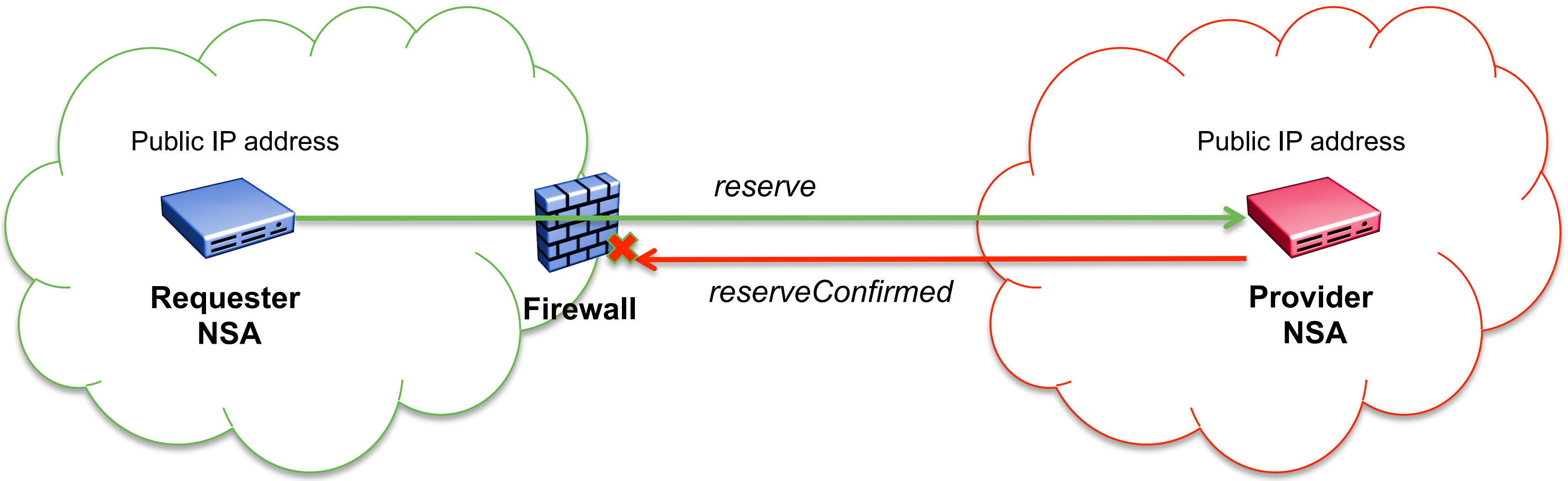


# The Firewall Problem



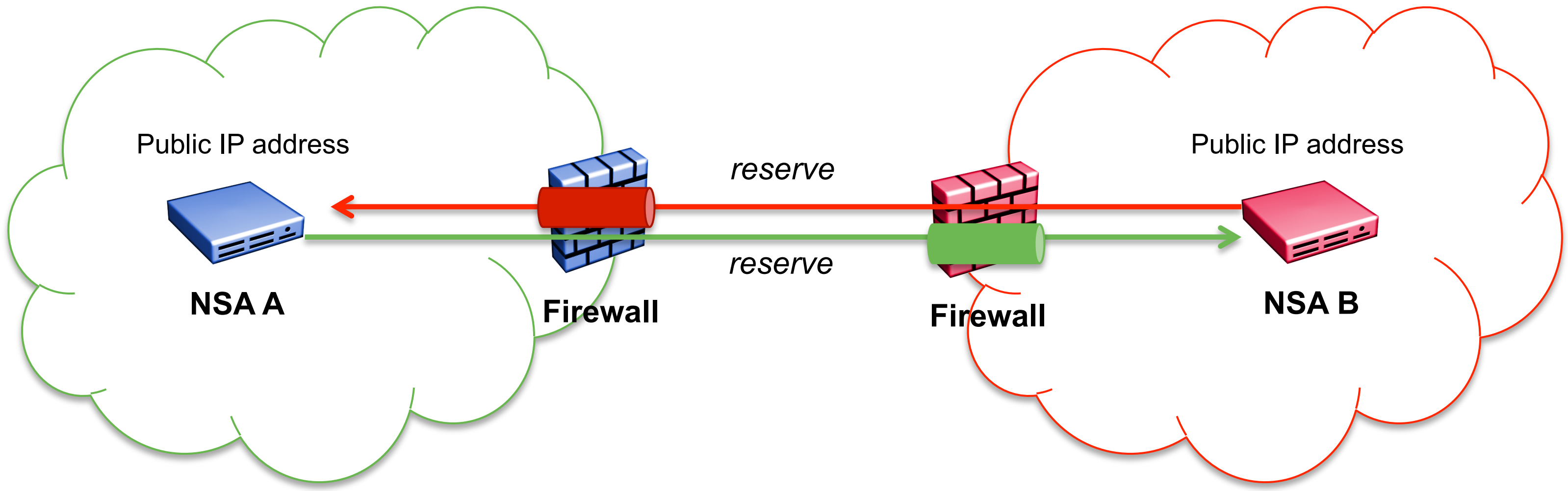
- Requester NSA composes an NSI reserve request message populating the "replyTo" field with its SOAP endpoint using private IP address for asynchronous response.
- Requester NSA behind firewall issues HTTP reserve request to provider NSA on the public network.
- Firewall NATs HTTP request and passes on to Provider but does not NAT the private IP address in "replyTo" since this is embedded in the SOAP message.
- Provider NSA cannot reach the private IP address to deliver the reserveConfirmed message.

# The Firewall Problem



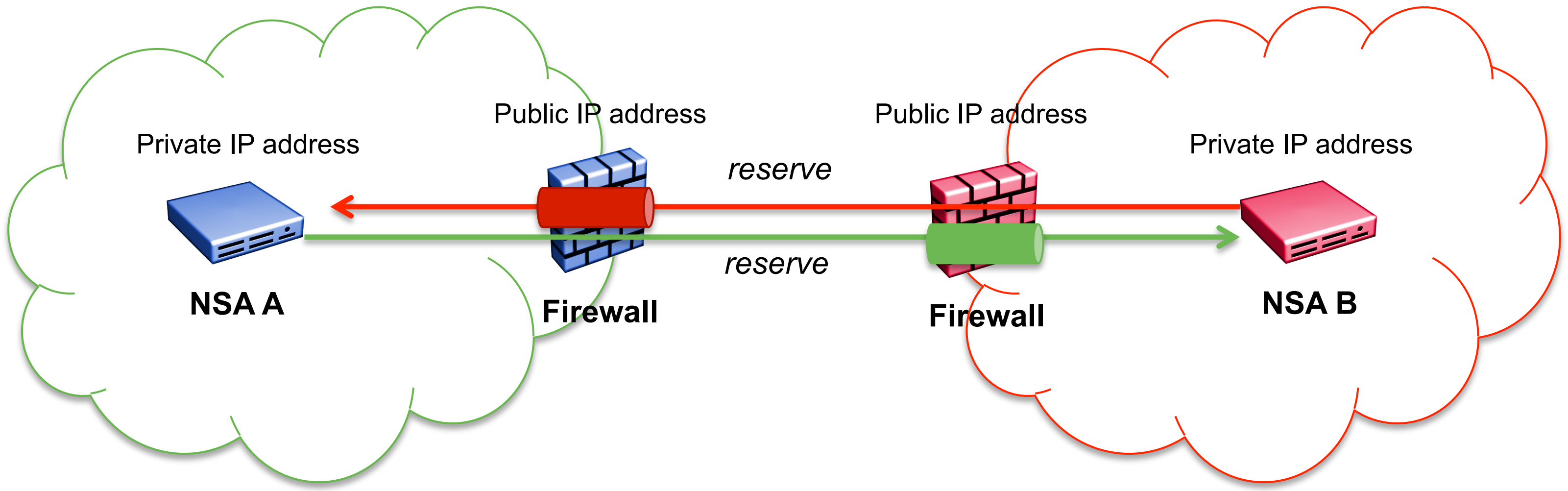
- Requester NSA composes an NSI reserve request message populating the “replyTo” field with its SOAP endpoint using public IP address for asynchronous response.
- Requester NSA behind firewall issues HTTP reserve request to provider NSA on the public network.
- Firewall passes request on to Provider but requires no NATing of addresses.
- Provider NSA cannot reach the public IP address of requester NSA to deliver the reserveConfirmed message as the firewall is blocking incoming HTTP connections.

# Proper Configuration of Firewall



- For NSA with public IP addresses behind the firewall:
    - Access control lists can be set for peer NSA in combination with port filtering to allow an NSA behind a firewall to have traffic to it's HTTP server port passed through.

# Proper Configuration of Firewall



- For NSA with private IP addresses behind the firewall:
    - Firewall will need to act as public entity for NSA.
    - Access control lists can be set for peer NSA in combination with NAT and port forwarding to allow the requesting NSA to be mapped through to the provider NSA's HTTP server port within the DMZ
    - A requesting NSA behind a firewall will need to provide the public facing IP address and port of the firewall within the replyTo field of the SOAP request.

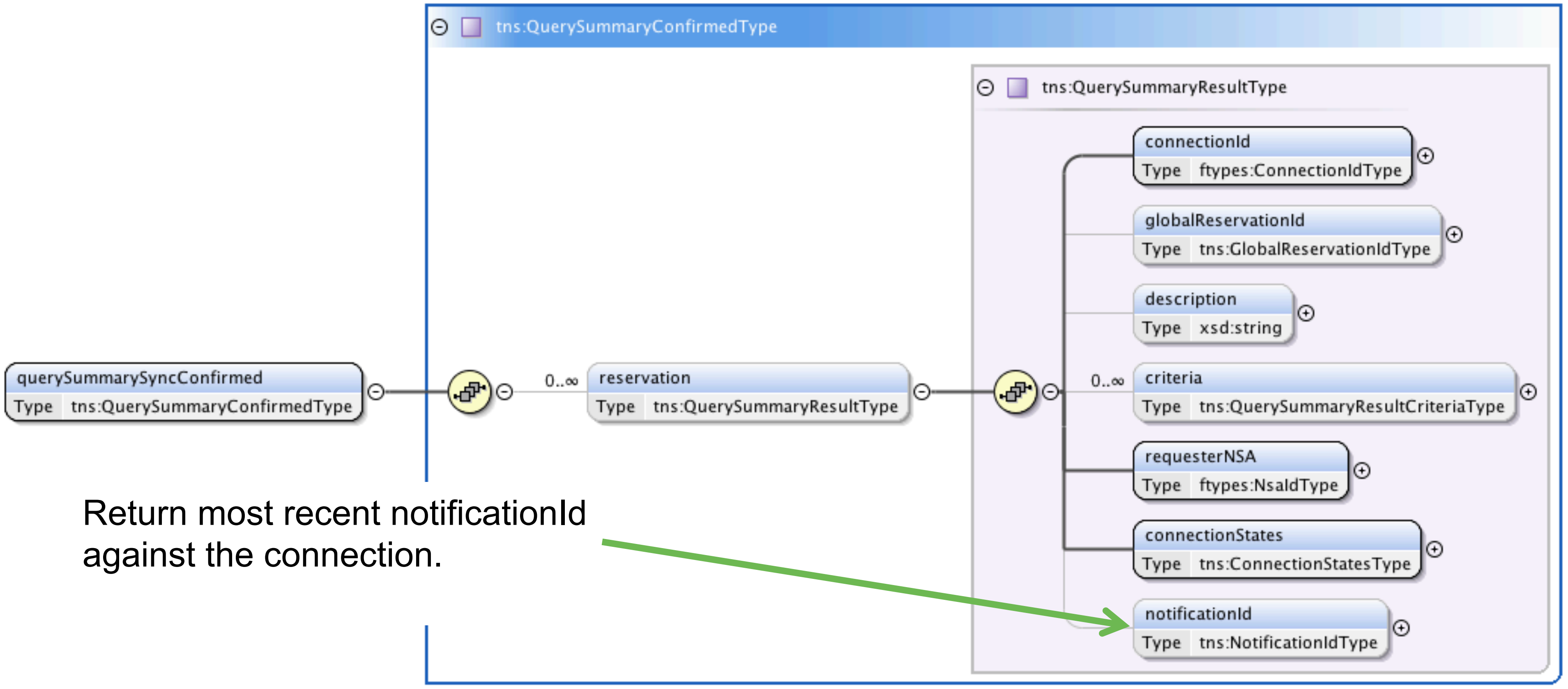
## But when all else fails…

- Build requester NSA using synchronous messaging model removing need for provider initiated communications
- Issue NSI CS operations without “replyTo” asynchronous callback specified
- Use querySummarySync() to observe state machine transitions, and therefore, operation status
- Retrieve pending notifications using queryNotificationSync()

## Notification Support

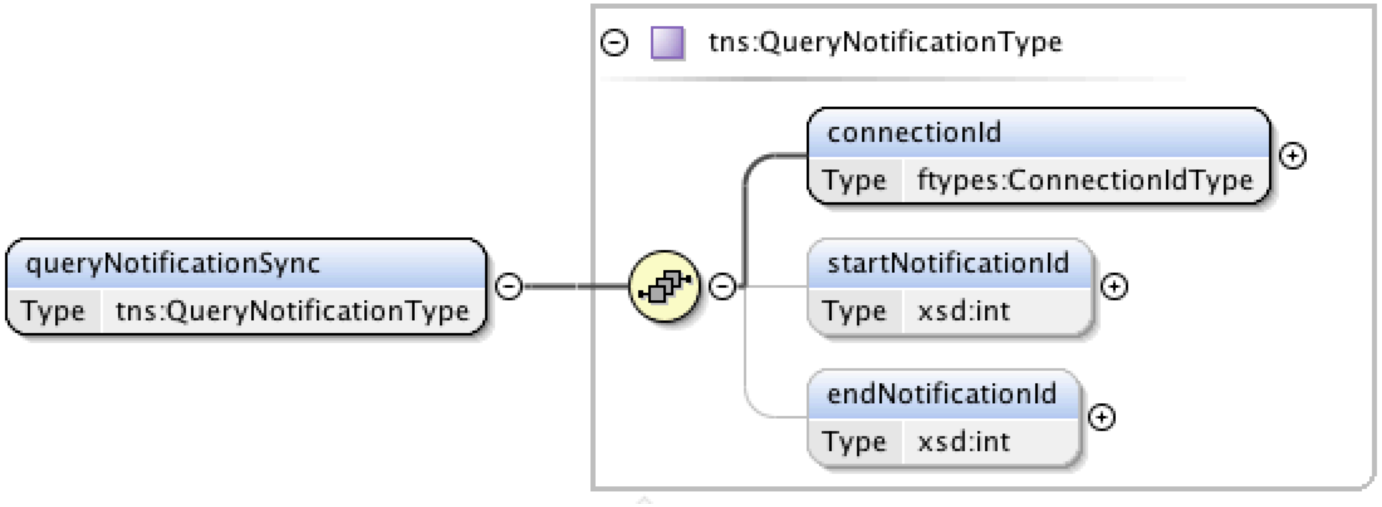
- Extend querySummary(Sync) message set to identify pending notifications.
- Introduce new queryNotificationSync message to retrieve list of notifications against a connection.
- We change the definition of a notificationId to be a linearly increasing integer unique in the context of a connectionId, removing the need for a separate ordering attribute.

## QuerySummaryConfirmedType


tns:QuerySummaryConfirmedType
querySummarySyncConfirmed
Type tns:QuerySummaryConfirmedType
0..∞
reservation
Type tns:QuerySummaryResultType
tns:QuerySummaryResultType
connectionId
Type ftypes:ConnectionIdType
globalReservationId
Type tns:GlobalReservationIdType
description
Type xsd:string
0..∞
criteria
Type tns:QuerySummaryResultCriteriaType
requesterNSA
Type ftypes:NsaIdType
connectionStates
Type tns:ConnectionStatesType
notificationId
Type tns:NotificationIdType

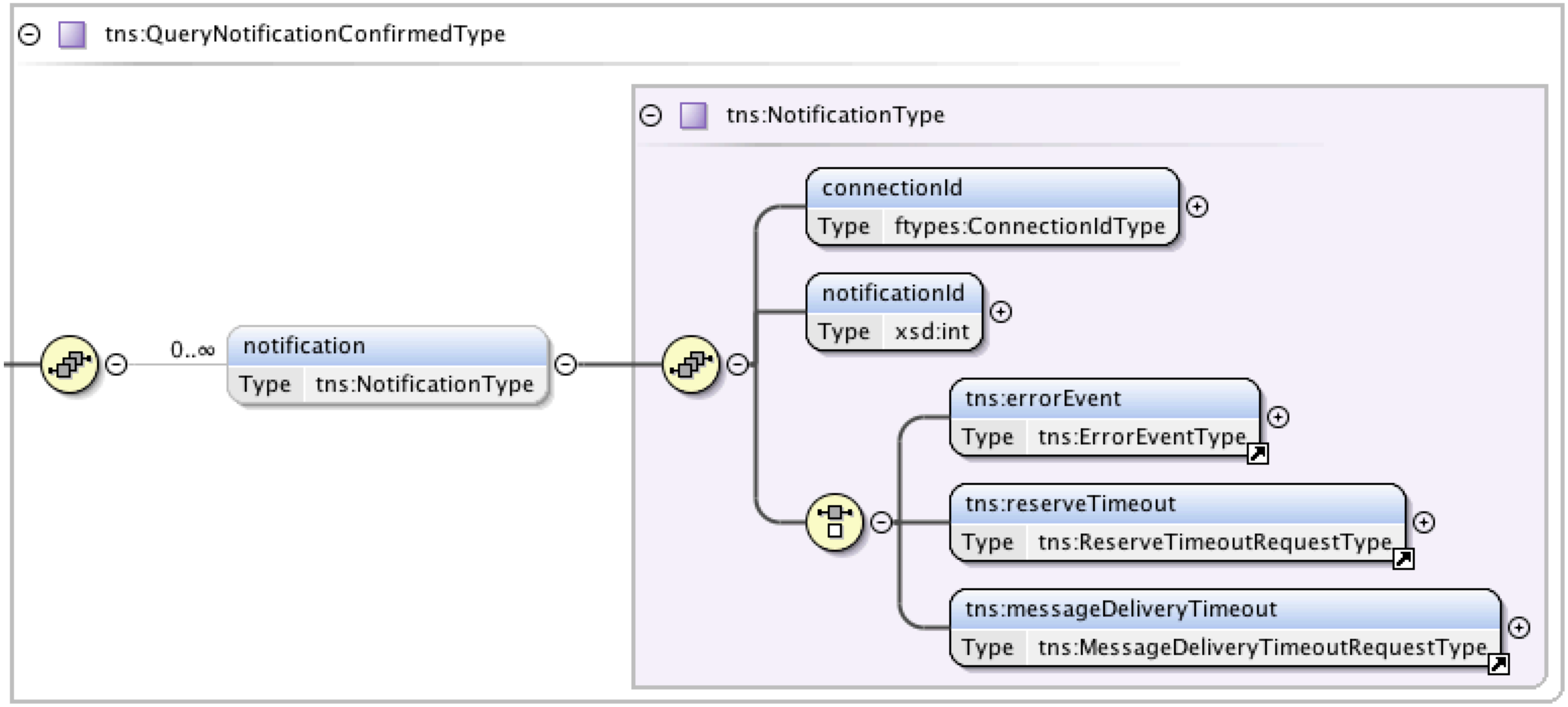
Return most recent notificationId against the connection.

# QueryNotificationType



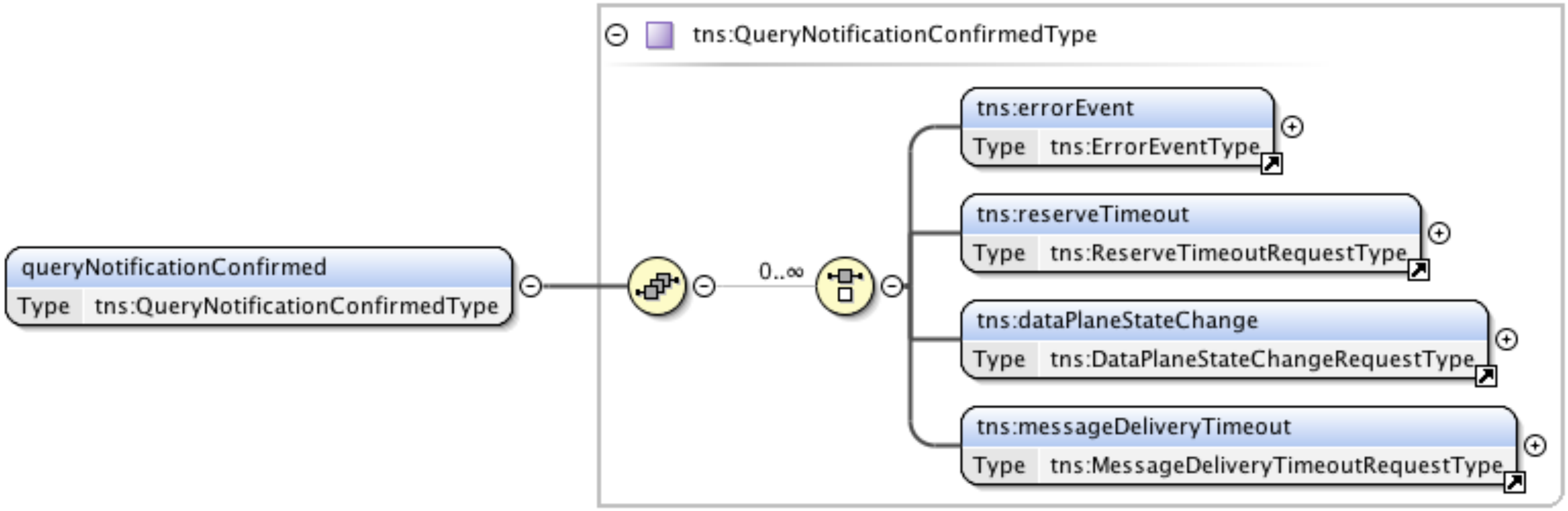
- Return a list of notifications in the context of a connectionId.
- startNotificationId – if present the query will return a list of notifications starting from this identifier. If not present then query will return notifications starting from oldest available.
- endNotificationId – if present the query will return notifications up to and including this identifier. If not present then query will return notifications up to and including the newest.

# QueryNotificationConfirmedType


tns:QueryNotificationConfirmedType
0..∞
notification
Type tns:NotificationType
tns:NotificationType
connectionId
Type ftypes:ConnectionIdType
notificationId
Type xsd:int
tns:errorEvent
Type tns:ErrorEventType
tns:reserveTimeout
Type tns:ReserveTimeoutRequestType
tns:messageDeliveryTimeout
Type tns:MessageDeliveryTimeoutRequestType

Originally contained just the notifications not impacting the state machine.

# The first change

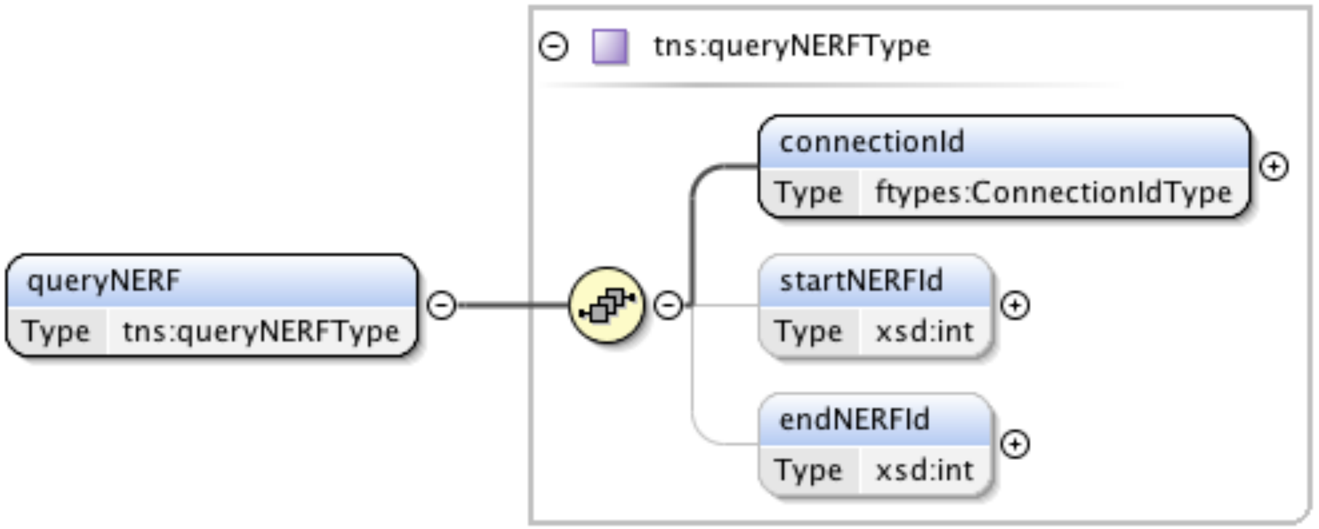

tns:QueryNotificationConfirmedType
queryNotificationConfirmed
Type tns:QueryNotificationConfirmedType
0..∞
tns:errorEvent
Type tns:ErrorEventType
tns:reserveTimeout
Type tns:ReserveTimeoutRequestType
tns:dataPlaneStateChange
Type tns:DataPlaneStateChangeRequestType
tns:messageDeliveryTimeout
Type tns:MessageDeliveryTimeoutRequestType

We added dataPlaneStateChange notifications for consistency in notification behaviors and moved the connectionId and notificationId into a NotificationBaseType that is part of every notification.

## Why a second change?

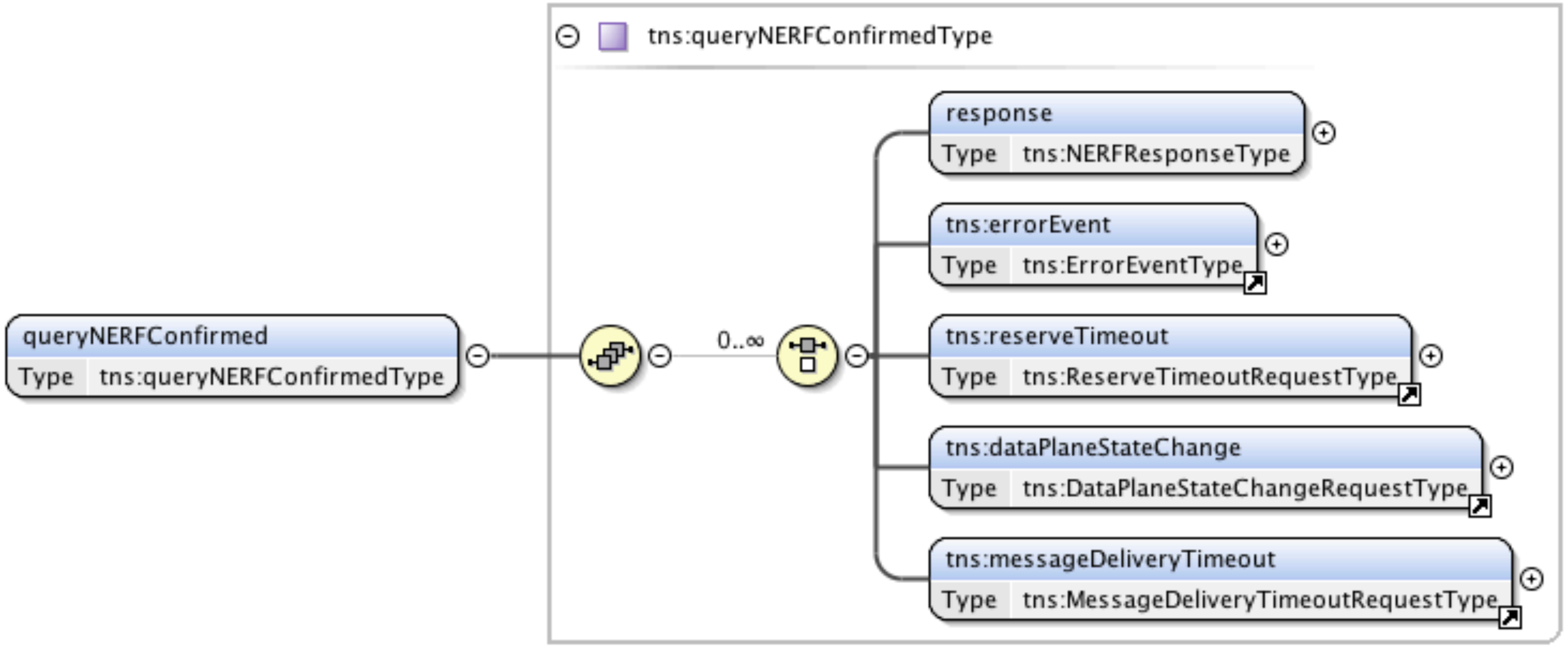
- We introduced the Error response to the NSI protocol reporting error conditions on request operations that did not impact the state machine, and therefore, did not use the Failed response mechanism.
- A polling client has no way to determine if an Error has occurred because it is not a notification, and does not impact the state machine.
- We agreed to add Error responses to this new queryNotification mechanism.
- Now that we are adding Error responses we might as well make it consistent and add all responses removing the need to observe state machine changes to determine if operations were successful.

## queryNERFType



The queryNERF message provides a mechanism for a Requester NSA to query a Provider NSA for a set of Notifications, Errors, Responses, and Failures (NERF) against a specific connectionId.

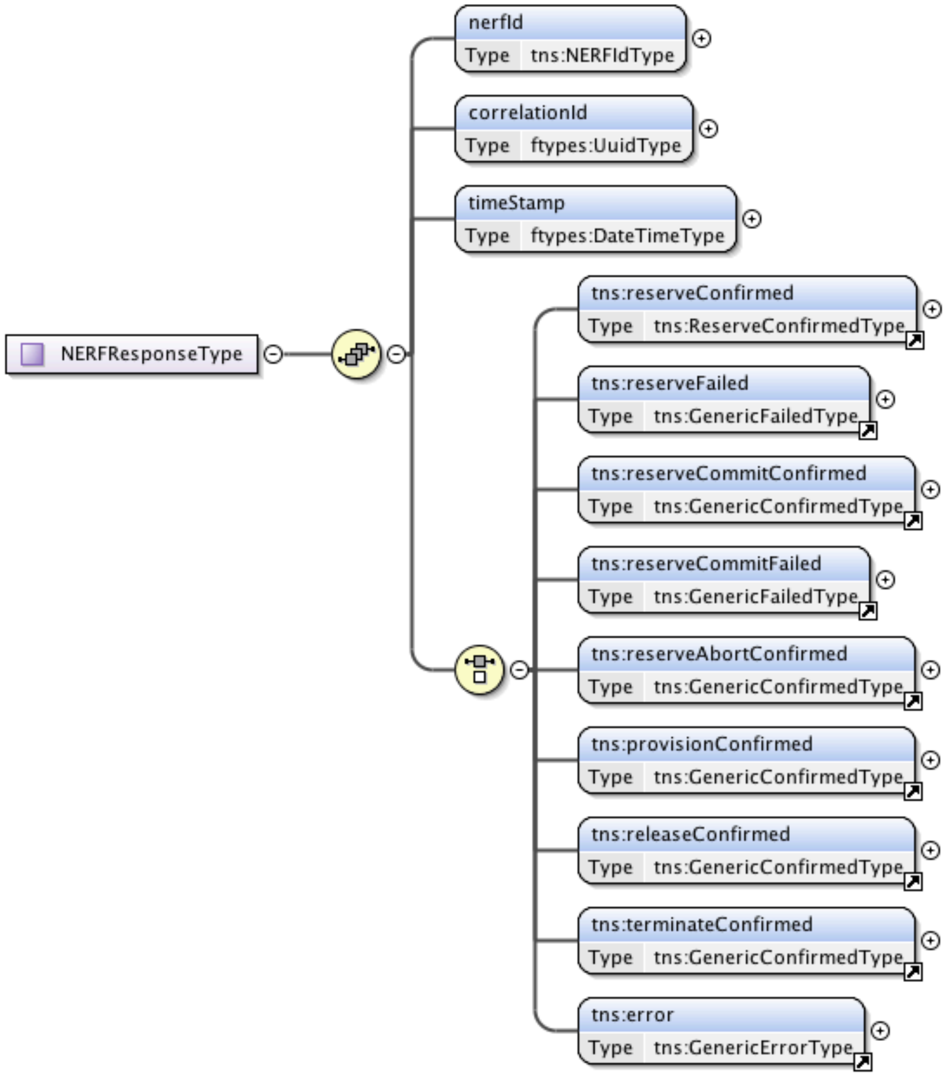
# queryNERFConfirmedType


tns:queryNERFConfirmedType
queryNERFConfirmed
Type tns:queryNERFConfirmedType
0..∞
response
Type tns:NERFResponseType
tns:errorEvent
Type tns:ErrorEventType
tns:reserveTimeout
Type tns:ReserveTimeoutRequestType
tns:dataPlaneStateChange
Type tns:DataPlaneStateChangeRequestType
tns:messageDeliveryTimeout
Type tns:MessageDeliveryTimeoutRequestType

We expanded the original notification confirmed type to include a response element encapsulating confirmed, failed, and error messages matching the specified query criteria.

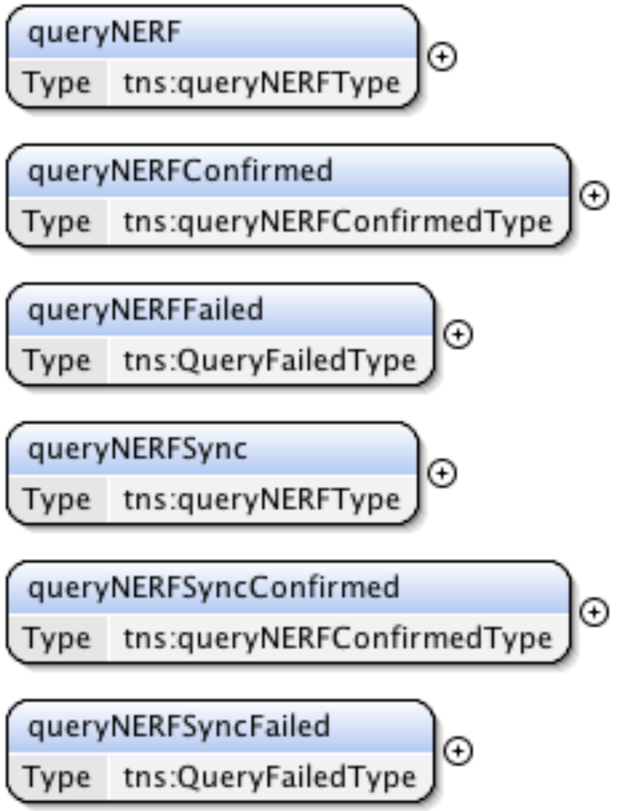
# NERFResponseType

Encapsulates the Confirmed, Failed, and Error response messages along with the original *correlationId* (from the NSI header), the per *connectionId* generated *nerfId* that is not part of response message, and a timestamp indicating when this response was received.

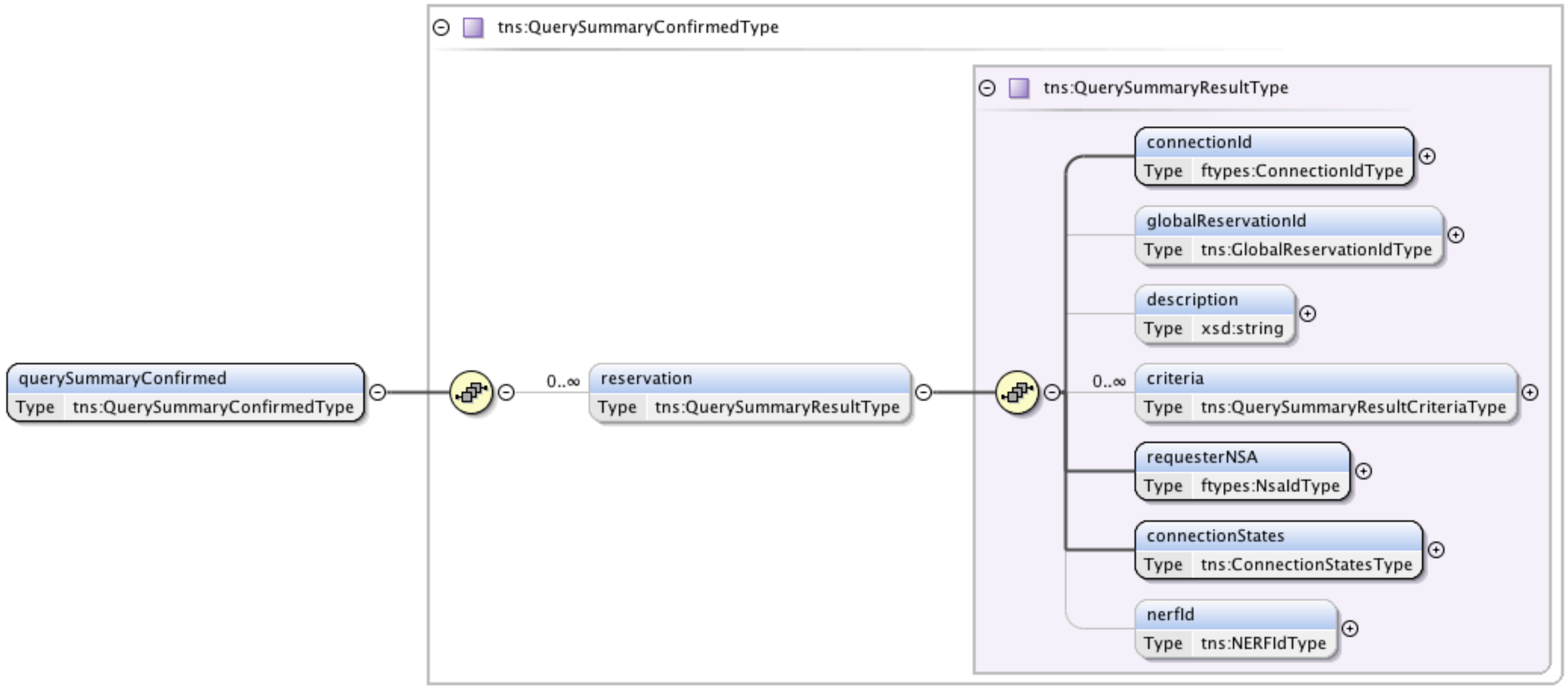

NERFResponseType
nerfId
Type tns:NERFIdType
correlationId
Type ftypes:UuidType
timeStamp
Type ftypes:DateTimeType
tns:reserveConfirmed
Type tns:ReserveConfirmedType
tns:reserveFailed
Type tns:GenericFailedType
tns:reserveCommitConfirmed
Type tns:GenericConfirmedType
tns:reserveCommitFailed
Type tns:GenericFailedType
tns:reserveAbortConfirmed
Type tns:GenericConfirmedType
tns:provisionConfirmed
Type tns:GenericConfirmedType
tns:releaseConfirmed
Type tns:GenericConfirmedType
tns:terminateConfirmed
Type tns:GenericConfirmedType
tns:error
Type tns:GenericErrorType

# Sync and Async message sets

We rename the queryNotification message sets to queryNERF to better describe the new functionality.



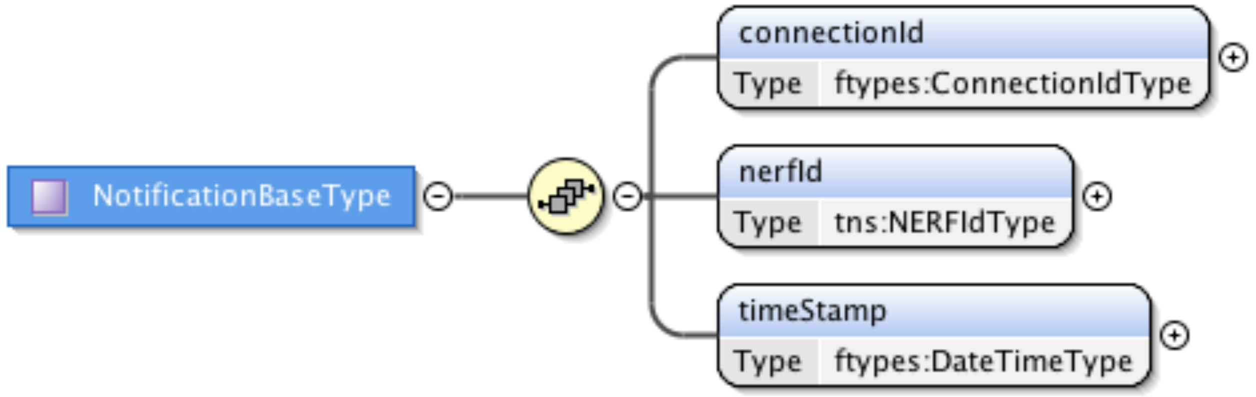
## query*Confirmed



tns:QuerySummaryConfirmedType
tns:QuerySummaryResultType
querySummaryConfirmed
Type tns:QuerySummaryConfirmedType
0..∞
reservation
Type tns:QuerySummaryResultType
connectionId
Type ftypes:ConnectionIdType
globalReservationId
Type tns:GlobalReservationIdType
description
Type xsd:string
0..∞
criteria
Type tns:QuerySummaryResultCriteriaType
requesterNSA
Type ftypes:NsaIdType
connectionStates
Type tns:ConnectionStatesType
nerfId
Type tns:NERFIdType

Modify the querySummaryConfirmed and queryRecursiveConfirmed messages to contain the nerfId instead of notificationId.

# NotificationBaseType



Modify the NotificationBaseType to rename notificationId to nerfId.