

Notification Message Headers and Bundles

draft-voit-netconf-notification-messages-01

Eric Voit, Andy Bierman, Alexander Clemm, Tim Jenkins

Functionality per Draft

Subscribed Notifications <ul style="list-style-type: none"> • Dynamic & Configured subscriptions • Multiple subscriptions / transport • Multiple configured receivers • Establish, modify, delete, kill RPC • State change notifications • Suspend/resume • Filtering full notifications 		<ul style="list-style-type: none"> • Stream discovery • Replay (and start time negotiation) • Prioritization • Monitoring / reporting • QoS • Error responses
YANG Datastore Push <ul style="list-style-type: none"> • Datastore on-change and periodic triggers • Filtering objects within a notification • Authorization model per object • Sending of full YANG trees or yang patch 		<ul style="list-style-type: none"> • Tagging of partial updates • Tagging of on-change object support • Negotiation of filters and period lengths • More error responses
YANG Notifications2 <ul style="list-style-type: none"> • Encapsulation Headers objects: Signature, de-duplication, severity, originator • Bundled records and record types 		
NETCONF Transport for Subscribed Notifications <ul style="list-style-type: none"> • Transport mapping 	RESTCONF & HTTP2 Transport for Subscribed Notifications <ul style="list-style-type: none"> • Transport mappings (including HTTP2 QoS) • Heartbeats and clean-up 	

Legend

draft-ietf-netconf-yang-push
draft-ietf-netconf-subscribed-notifications
draft-voit-netmod-yang-notifications2
draft-ietf-netconf-event-netconf
draft-ietf-netconf-event-restconf

Purpose

- Create a way to “bundle” multiple update and notification records into a single notification message
- Allow to combine updates from multiple subscriptions into a single notification
- Distinguish between what goes into the “record”, and what goes into the “bundle”
 - Record contents distinguished from wrapper that it is contained in
 - Wrapper can be single notification message (one record), or bundled-notification message (multiple records)

#1 Transport Agnostic Header Objects

```
+---n notification-message
  +---ro notification-message-header
  | +---ro record-time
  | +---ro record-type?
  | +---ro subscription-id*
  | +---ro record-id?
  | +---ro observation-domain-id?
  | +---ro notification-id?
  | +---ro notification-time?
  | +---ro previous-notification-id?
  | +---ro dscp?
  | +---ro message-generator-id?
  | +---ro signature?
  +---ro receiver-record-contents?
```

*To deal with the possibility of distributed subscriptions involving notifications/streams from multiple line cards.
Touch points w/ draft-zheng-netconf-udp-pub-channel-00*

#2 bundling multiple notifications into a single transportable message

```
+---n bundled-notification-message
  +--ro bundled-notification-message-header
  |
  | +--ro notification-id?
  | +--ro notification-time
  | +--ro previous-notification-id?
  | +--ro dscp?
  | +--ro message-generator-id?
  | +--ro signature?
  | +--ro record-count?
  +--ro notification-records*
  +--ro notification-record-header
  |
  | +--ro record-time
  | +--ro record-type?
  | +--ro subscription-id*
  | +--ro record-id?
  | +--ro observation-domain-id?
  +--ro receiver-record-contents?
```

Next steps

- Refinements to data model
 - Some fields are overspecified and may not be needed → cleanup
e.g. various time stamps
- Synchronization/alignment with the concept of multiple message generators for a subscription
 - Separation of concerns: decoupling bundling, common headers, and loss detection
- Ask for WG adoption