RPKI MIB modules

draft-ymbk-rpki-rtr-mib-02.txt

Bert Wijnen
Randy Bush
Keyur Patel
Michael Bauer
Changes since IETF81

• Merged the two documents:
  - RPKI-ORIGIN-VALIDATION-MIB
  - RPKI-ROUTER-MIB
  - into RPKI-ROUTER-MIB

• Addressed comments discussed at IETF81

• Review/Addressed comments from editors

• We were too late to submit it as WG doc.
Now in RPKI-ROUTER-MIB

- Defines Textual Convention ConnectionType
- lists all connections to the RPKI cache servers
- and the attributes for that connection
- lists all the origins (prefixOriginTable)
- defines 2 notifications to inform NMS about
  - connection(s) going up/down
  - entries to go stale
Todo in BGP4 MIB

• Todo: Extend BGP4 MIB, which one?
  - RFC 4273
  - or the draft that is still in the WG

• Basically, add one object to indicate the status to be valid, invalid or unknown

• possibly some counters as to how many of each of those we have
RpkiRtrConnectionType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION "The connection type or transport security suite (transport plus security mechanism) used between a router (as a client) and a cache server.

The following types have been defined in RFCnnnn:
ssh(1)   - sect 7.1, see also RFC4252.
tls(2)   - sect 7.2, see also RFC5246.
tcpMD5(3) - sect 7.3, see also RFC2385.
tcpAO(4) - sect 7.4, see also RFC5925.
tcp(5)   - sect 7.
ipse(6)  - sect 7, see also RFC4301.
other(7) - non of the above"

REFERENCE "The RPKI/Rtr Protocol, RFCnnnn - section 7"
RPKI Rtr Cache Server Table 1/2

RpkiRtrCacheServerTableEntry ::= SEQUENCE {
    rpkiRtrCacheServerAddressType      InetAddressType,
    rpkiRtrCacheServerRemoteAddress    InetAddress,
    rpkiRtrCacheServerRemotePort       InetPortNumber,
    rpkiRtrCacheServerLocalAddress     InetAddress,
    rpkiRtrCacheServerLocalPort        netPortNumber,
    rpkiRtrCacheServerPreference       Unsigned32,
    rpkiRtrCacheServerConnectionType   RpkiRtrConnectionType,
    rpkiRtrCacheServerConnectionStatus INTEGER,
    rpkiRtrCacheServerDescription      LongUtf8String,
    rpkiRtrCacheServerMsgsReceived     Counter32,
    rpkiRtrCacheServerMsgsSent         Counter32,
    rpkiRtrCacheServerV4ActiveRecords  Gauge32,
    rpkiRtrCacheServerV4Announcements  Counter32,
    rpkiRtrCacheServerV4Withdrawals    Counter32,
}
### RPKI Rtr Cache Server Table 2/2

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpkiRtrCacheServerV6ActiveRecords</td>
<td>Gauge32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerV6Announcements</td>
<td>Counter32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerV6Withdrawals</td>
<td>Counter32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerLatestSerial</td>
<td>Unsigned32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerNonce</td>
<td>Unsigned32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerRefreshTimer</td>
<td>Unsigned32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerTimeToRefresh</td>
<td>Integer32</td>
</tr>
<tr>
<td>rpkiRtrCacheServerId</td>
<td>Unsigned32</td>
</tr>
</tbody>
</table>
### RPKI Rtr Cache Server Errors Table

rpkiRtrCacheServerErrorsTableEntry OBJECT-TYPE
  AUGMENTS { rpkiRtrCacheServerTableEntry } 

RpkiRtrCacheServerErrorsTableEntry ::= SEQUENCE { 
  rpkiRtrCacheServerErrorsCorruptData       Counter32, 
  rpkiRtrCacheServerErrorsInternalError     Counter32, 
  rpkiRtrCacheServerErrorsNoData            Counter32, 
  rpkiRtrCacheServerErrorsInvalidRequest    Counter32, 
  rpkiRtrCacheServerErrorsUnsupportedVersion Counter32, 
  rpkiRtrCacheServerErrorsUnsupportedPdu     Counter32, 
  rpkiRtrCacheServerErrorsWithdrawalUnknown Counter32, 
  rpkiRtrCacheServerErrorsDuplicateAnnounce Counter32 
}
RPKI Rtr Prefix Origin Table

rpkiRtrPrefixOriginTableEntry OBJECT-TYPE
  INDEX
  
  RpkiRtrPrefixOriginTableEntry ::= SEQUENCE {
    rpkiRtrPrefixOriginAddressType InetAddressType,
    rpkiRtrPrefixOriginAddress InetAddress,
    rpkiRtrPrefixOriginMinLength InetAddressPrefixLength,
    rpkiRtrPrefixOriginMaxLength InetAddressPrefixLength,
    rpkiRtrPrefixOriginASN InetAutonomousSystemNumber,
    rpkiRtrPrefixOriginCacheServerId Unsigned32
  }

Bert Wijnen, SIDR WG at IETF82, 15 Nov 2011
Making sure we have a unique INDEX.
Is adding ASN sufficient?

From Rob Austein:
I'm not sure that even adding the ASN to this makes the index unique. I seem to recall a SIDR WG discussion in which Róbert Kisteleki convinced me that we had to allow overlapping ROAs. I haven't quite figured out how that maps to this discussion, but I suspect it means that we'd have to add both ASN and maxLength to the index to make it unique.
RPKI Rtr Cache Server Conn Notifications

```
rpkiRtrCacheServerConnectionStateChange NOTIFICATION-TYPE
   OBJECTS { rpkiRtrCacheServerConnectionStatus,
               rpkiRtrCacheServerLatestSerial,
               rpkiRtrCacheServerNonce
          }

rpkiRtrCacheServerConnectionToGoStale NOTIFICATION-TYPE
   OBJECTS { rpkiRtrCacheServerV4ActiveRecords,
               rpkiRtrCacheServerV6ActiveRecords,
               rpkiRtrCacheServerLatestSerial,
               rpkiRtrCacheServerNonce,
               rpkiRtrCacheServerRefreshTimer,
               rpkiRtrCacheServerTimeToRefresh
          }
```
Next steps

• make it a WG doc
• Get WG review/comments/input
• Do we want to add stuff to the bgp4-MIB
  - RFC 4273
  - or the draft that is still in the WG
  - MUST have/get WG input on that before I spend time working on this