MODULE *gNMI*

LOCAL INSTANCE *Naturals*

LOCAL INSTANCE *Sequences*

LOCAL INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

CONSTANT *Nil*

CONSTANT *OK*, *Error*

VARIABLE *conns*

$gRPC \triangleq$ INSTANCE *gRPC* WITH
OK $\leftarrow$ "OK",
Error $\leftarrow$ "Error"

$vars \triangleq \langle conns \rangle$

MODULE *Messages*

The *Messages* module defines predicates for receiving, sending, and verifying all the messages supported by *gNMI*.

Message type constants

CONSTANT
CapabilityRequest,
CapabilityResponse

CONSTANTS
GetRequest,
GetResponse

CONSTANTS
SetRequest,
SetResponse

CONSTANTS
SubscribeRequest,
SubscribeResponse

CONSTANTS
Invalid,
Delete,
Replace,
Update

LOCAL $messageTypes \triangleq$
{*CapabilityRequest*,

$CapabilityResponse$,
$GetRequest$,
$GetResponse$,
$SetRequest$,
$SetResponse$,
$SubscribeRequest$,
$SubscribeResponse\}$

Message types should be defined as strings to simplify debugging

ASSUME $\forall\, m \in messageTypes : m \in$ STRING

LOCAL $opTypes \triangleq$
$\{Invalid$,
$Delete$,
$Replace$,
$Update\}$

Operation types should be defined as strings to simplify debugging

ASSUME $\forall\, m \in opTypes : m \in$ STRING

This section defines predicates for identifying *gNMI* message types on the network.

$IsCapabilityRequest(m) \triangleq m.type = CapabilityRequest$

$IsCapabilityResponse(m) \triangleq m.type = CapabilityResponse$

$IsGetRequest(m) \triangleq m.type = GetRequest$

$IsGetResponse(m) \triangleq m.type = GetResponse$

$IsSetRequest(m) \triangleq m.type = SetRequest$

$IsSetResponse(m) \triangleq m.type = SetResponse$

$IsSubscribeRequest(m) \triangleq m.type = SubscribeRequest$

$IsSubscribeResponse(m) \triangleq m.type = SubscribeResponse$

This section defines predicates for validating *gNMI* message contents. The predicates provide precise documentation on the *gNMI* message format and are used within the spec to verify that steps adhere to the *gNMI* protocol specification.

LOCAL $ValidFields(m, fs) \triangleq$
$\land \quad \{f : f \in$ DOMAIN $m\} \circ fs \neq \{\}$

LOCAL $ValidUpdate(m) \triangleq$
$\land \quad ValidFields(m, \{$"path", "value"$\})$

```
LOCAL ValidUpdates(ms) ≜
    ∧  ∀ m ∈ ms : ValidUpdate(m)

LOCAL ValidDelete(m) ≜
    ∧  ValidFields(m, {"path"})

LOCAL ValidDeletes(ms) ≜
    ∧  ∀ m ∈ ms : ValidDelete(m)

LOCAL ValidNotification(m) ≜
    ∧  ValidFields(m, {"prefix"})
    ∧  "update" ∈ DOMAIN m ⇒ ValidUpdates(m["update"])
    ∧  "delete" ∈ DOMAIN m ⇒ ValidDeletes(m["delete"])

LOCAL ValidCapabilityRequest(m) ≜ TRUE

LOCAL ValidCapabilityResponse(m) ≜ TRUE

LOCAL ValidGetRequest(m) ≜
    ∧  ValidFields(m, {"path"})

LOCAL ValidGetResponse(m) ≜
    ∧  ValidFields(m, {"notification"})
    ∧  ValidNotification(m["notification"])

LOCAL ValidSetRequest(m) ≜
    ∧  ValidFields(m, {"prefix"})
    ∧  "update" ∈ DOMAIN m ⇒ ValidUpdates(m["update"])
    ∧  "replace" ∈ DOMAIN m ⇒ ValidUpdates(m["replace"])
    ∧  "delete" ∈ DOMAIN m ⇒ ValidDeletes(m["delete"])

LOCAL ValidOperation(op) ≜
    ∧  op ∈ {Invalid, Delete, Replace, Update}

LOCAL ValidResult(m) ≜
    ∧  ValidFields(m, {"path", "op"})
    ∧  ValidOperation(m["op"])

LOCAL ValidSetResponse(m) ≜
    ∧  ValidFields(m, {"prefix", "response"})
    ∧  ValidResult(m["response"])

LOCAL ValidSubscription(m) ≜
    ∧  ValidFields(m, {"path", "mode"})

LOCAL ValidSubscribeRequest(m) ≜
    ∨  ∧ "subscribe" ∈ DOMAIN m
       ∧ ValidFields(m["subscribe"], {"prefix", "subscription"})
       ∧ ValidSubscription(m["subscribe"]["subscription"])
```

```
    ∨   ∧ "poll" ∈ DOMAIN m
    ∨   ∧ "aliases" ∈ DOMAIN m
        ∧ ∀ a ∈ m["aliases"] : ValidFields(a, {"path", "alias"})

LOCAL ValidSubscribeResponse(m) ≜
    ∨   ∧ "update" ∈ DOMAIN m
        ∧ ValidNotification(m["update"])
```

This section defines operators for constructing *gNMI* messages.

```
LOCAL SetType(m, t) ≜ [m EXCEPT !.type = t]

WithCapabilityRequest(m) ≜
  IF Assert(ValidCapabilityRequest(m), "Invalid CapabilityRequest")
   THEN SetType(m, CapabilityRequest)
   ELSE Nil

WithCapabilityResponse(m) ≜
  IF Assert(ValidCapabilityResponse(m), "Invalid CapabilityResponse")
   THEN SetType(m, CapabilityResponse)
   ELSE Nil

WithGetRequest(m) ≜
  IF Assert(ValidGetRequest(m), "Invalid GetRequest")
   THEN SetType(m, GetRequest)
   ELSE Nil

WithGetResponse(m) ≜
  IF Assert(ValidGetResponse(m), "Invalid GetResponse")
   THEN SetType(m, GetResponse)
   ELSE Nil

WithSetRequest(m) ≜
  IF Assert(ValidSetRequest(m), "Invalid SetRequest")
   THEN SetType(m, SetRequest)
   ELSE Nil

WithSetResponse(m) ≜
  IF Assert(ValidSetResponse(m), "Invalid SetResponse")
   THEN SetType(m, SetResponse)
   ELSE Nil

WithSubscribeRequest(m) ≜
  IF Assert(ValidSubscribeRequest(m), "Invalid SubscribeRequest")
   THEN SetType(m, SubscribeRequest)
   ELSE Nil
```

$WithSubscribeResponse(m) \triangleq$
IF $Assert(ValidSubscribeResponse(m), \text{"Invalid SubscribeResponse"})$
THEN $SetType(m, SubscribeResponse)$
ELSE Nil

The *Messages* module is instantiated locally to avoid access from outside the module.

LOCAL $Messages \triangleq$ INSTANCE $Messages$ WITH
$CapabilityRequest \leftarrow$ "CapabilityRequest",
$CapabilityResponse \leftarrow$ "CapabilityResponse",
$GetRequest \leftarrow$ "GetRequest",
$GetResponse \leftarrow$ "GetResponse",
$SetRequest \leftarrow$ "SetRequest",
$SetResponse \leftarrow$ "SetResponse",
$SubscribeRequest \leftarrow$ "SubscribeRequest",
$SubscribeResponse \leftarrow$ "SubscribeResponse",
$Invalid \leftarrow$ "Invalid",
$Delete \leftarrow$ "Delete",
$Replace \leftarrow$ "Replace",
$Update \leftarrow$ "Update"

MODULE *Client*

The *Client* module provides operators for managing and operating on *gNMI* client connections and specifies the message types supported for the client.

MODULE *Send*

This module provides message type operators for the message types that can be send by the *gNMI* client.

$CapabilityRequest(c, m) \triangleq$
$\land\ gRPC!Client!Send(c, Messages!WithCapabilityRequest(m))$

$GetRequest(c, m) \triangleq$
$\land\ gRPC!Client!Send(c, Messages!WithGetRequest(m))$

$SetRequest(c, m) \triangleq$
$\land\ gRPC!Client!Send(c, Messages!WithSetRequest(m))$

$SubscribeRequest(c, m) \triangleq$
$\land\ gRPC!Client!Send(c, Messages!WithSubscribeRequest(m))$

Instantiate the $gNMI!Client!$Requests module
$Send \triangleq$ INSTANCE $Send$

MODULE *Receive*

This module provides predicates for the types of messages that can be received by an *gNMI* client.

$CapabilityResponse(c,\ h(\_)) \triangleq$
$\quad gRPC!Client!Handle(c,\ \text{LAMBDA}\ x,\ m :$
$\quad\quad \land\ Messages!IsCapabilityResponse(m)$
$\quad\quad \land\ gRPC!Client!Receive(c)$
$\quad\quad \land\ h(m))$

$GetResponse(c,\ h(\_)) \triangleq$
$\quad gRPC!Client!Handle(c,\ \text{LAMBDA}\ x,\ m :$
$\quad\quad \land\ Messages!IsGetResponse(m)$
$\quad\quad \land\ gRPC!Client!Receive(c)$
$\quad\quad \land\ h(m))$

$SetResponse(c,\ h(\_)) \triangleq$
$\quad gRPC!Client!Handle(c,\ \text{LAMBDA}\ x,\ m :$
$\quad\quad \land\ Messages!IsSetResponse(m)$
$\quad\quad \land\ gRPC!Client!Receive(c)$
$\quad\quad \land\ h(m))$

$SubscribeResponse(c,\ h(\_)) \triangleq$
$\quad gRPC!Client!Handle(c,\ \text{LAMBDA}\ x,\ m :$
$\quad\quad \land\ Messages!IsSubscribeResponse(m)$
$\quad\quad \land\ gRPC!Client!Receive(c)$
$\quad\quad \land\ h(m))$

Instantiate the *gNMI*!*Client*!Responses module

$Handle \triangleq \text{INSTANCE}\ Receive$

$Connect(s,\ d) \triangleq gRPC!Client!Connect(s,\ d)$

$Disconnect(c) \triangleq gRPC!Client!Disconnect(c)$

Provides operators for the *gNMI* client

$Client \triangleq \text{INSTANCE}\ Client$

MODULE *Server*

The *Server* module provides operators for managing and operating on *gNMI* servers and specifies the message types supported for the server.

MODULE *Send*

This module provides message type operators for the message types that can be send by the *gNMI* server.

$CapabilityResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithCapabilityResponse(m))$

$GetResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithGetResponse(m))$

$SetResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithSetResponse(m))$

$SubscribeResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithSubscribeResponse(m))$

Instantiate the $gNMI!Server!$Responses module
$Send \triangleq$ INSTANCE $Send$

MODULE *Reply*

This module provides message type operators for the message types that can be send by the *gNMI* server.

$CapabilityResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithCapabilityResponse(m))$

$GetResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithGetResponse(m))$

$SetResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithSetResponse(m))$

$SubscribeResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithSubscribeResponse(m))$

Instantiate the $gNMI!Server!Reply$ module
$Reply \triangleq$ INSTANCE $Reply$

MODULE *Receive*

This module provides predicates for the types of messages that can be received by an *gNMI* server.

$CapabilityRequest(c,\ h(\_)) \triangleq$
$\quad gRPC!Server!Handle(c,$ LAMBDA $x,\ m :$
$\quad\quad \land\ Messages!IsCapabilityRequest(m)$
$\quad\quad \land\ gRPC!Server!Receive(c)$
$\quad\quad \land\ h(m))$

$GetRequest(c,\ h(\_)) \triangleq$
$\quad gRPC!Server!Handle(c,$ LAMBDA $x,\ m :$

$\wedge\ Messages!IsGetRequest(m)$
$\wedge\ gRPC!Server!Receive(c)$
$\wedge\ h(m))$

$SetRequest(c,\ h(\_)) \triangleq$
$gRPC!Server!Handle(c,$ LAMBDA $x,\ m :$
$\wedge\ Messages!IsSetRequest(m)$
$\wedge\ gRPC!Server!Receive(c)$
$\wedge\ h(m))$

$SubscribeRequest(c,\ h(\_)) \triangleq$
$gRPC!Server!Handle(c,$ LAMBDA $x,\ m :$
$\wedge\ Messages!IsSubscribeRequest(m)$
$\wedge\ gRPC!Server!Receive(c)$
$\wedge\ h(m))$

Instantiate the $gNMI!Server!$Requests module
$Handle \triangleq$ INSTANCE $Receive$

Provides operators for the *gNMI* server
$Server \triangleq$ INSTANCE $Server$

The set of all open *gNMI* connections
$Connections \triangleq gRPC!Connections$

$Init \triangleq$
$\wedge\ gRPC!Init$

$Next \triangleq$
$\wedge\ gRPC!Next$

---

* Modification History
* Last modified *Wed Jan* 12 00:36:00 *PST* 2022 by *jordanhalterman*
* Created *Tue Jan* 11 23:46:02 *PST* 2022 by *jordanhalterman*