MODULE *TopoService*

The *TopoService* module provides a formal specification of the *ONOS* topology service. The spec defines the client and server interfaces for *ONOS Topo* and provides helpers for managing and operating on connections.

CONSTANT *Nil*

VARIABLE *conns*

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

LOCAL INSTANCE TLC

$vars \triangleq \langle conns \rangle$

MODULE *Messages*

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

Message type constants

CONSTANT
  CreateRequest,
  CreateResponse
CONSTANTS
  UpdateRequest,
  UpdateResponse
CONSTANTS
  DeleteRequest,
  DeleteResponse
CONSTANT
  GetRequest,
  GetResponse
CONSTANT
  ListRequest,
  ListResponse
CONSTANT
  WatchRequest,
  WatchResponse

LOCAL $messageTypes \triangleq$
  {*CreateRequest*,
   CreateResponse,
   UpdateRequest,
   UpdateResponse,
   DeleteRequest,
   DeleteResponse,

$GetRequest$,
$GetResponse$,
$ListRequest$,
$ListResponse$,
$WatchRequest$,
$WatchResponse\}$

Message types should be defined as strings to simplify debugging

ASSUME $\forall\, m \in messageTypes : m \in \text{STRING}$

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

$IsCreateRequest(m) \triangleq m.type = CreateRequest$

$IsCreateResponse(m) \triangleq m.type = CreateResponse$

$IsUpdateRequest(m) \triangleq m.type = UpdateRequest$

$IsUpdateResponse(m) \triangleq m.type = UpdateResponse$

$IsDeleteRequest(m) \triangleq m.type = DeleteRequest$

$IsDeleteResponse(m) \triangleq m.type = DeleteResponse$

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

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

$IsListRequest(m) \triangleq m.type = ListRequest$

$IsListResponse(m) \triangleq m.type = ListResponse$

$IsWatchRequest(m) \triangleq m.type = WatchRequest$

$IsWatchResponse(m) \triangleq m.type = WatchResponse$

This section defines predicates for validating *ONOS Topo* message contents. The predicates provide precise documentation on the $E2AP$ message format and are used within the spec to verify that steps adhere to the $E2AP$ protocol specification.

LOCAL $ValidCreateRequest(m) \triangleq$ TRUE

LOCAL $ValidCreateResponse(m) \triangleq$ TRUE

LOCAL $ValidUpdateRequest(m) \triangleq$ TRUE

LOCAL $ValidUpdateResponse(m) \triangleq$ TRUE

LOCAL $ValidDeleteRequest(m) \triangleq$ TRUE

LOCAL $ValidDeleteResponse(m) \triangleq$ TRUE

LOCAL $ValidGetRequest(m) \triangleq$ TRUE

LOCAL $ValidGetResponse(m) \triangleq$ TRUE

LOCAL $ValidListRequest(m) \triangleq$ TRUE

LOCAL $ValidListResponse(m) \triangleq$ TRUE

LOCAL $ValidWatchRequest(m) \triangleq$ TRUE

LOCAL $ValidWatchResponse(m) \triangleq$ TRUE

This section defines operators for constructing *ONOS Topo* messages.

LOCAL $SetType(m,\ t) \triangleq [m \text{ EXCEPT } !.type = t]$

$WithCreateRequest(m) \triangleq$
  IF $Assert(ValidCreateRequest(m),$ "Invalid CreateRequest")
  THEN $SetType(m,\ CreateRequest)$
  ELSE Nil

$WithCreateResponse(m) \triangleq$
  IF $Assert(ValidCreateResponse(m),$ "Invalid CreateResponse")
  THEN $SetType(m,\ CreateResponse)$
  ELSE Nil

$WithUpdateRequest(m) \triangleq$
  IF $Assert(ValidUpdateRequest(m),$ "Invalid UpdateRequest")
  THEN $SetType(m,\ UpdateRequest)$
  ELSE Nil

$WithUpdateResponse(m) \triangleq$
  IF $Assert(ValidUpdateResponse(m),$ "Invalid UpdateResponse")
  THEN $SetType(m,\ UpdateResponse)$
  ELSE Nil

$WithDeleteRequest(m) \triangleq$
  IF $Assert(ValidDeleteRequest(m),$ "Invalid DeleteRequest")
  THEN $SetType(m,\ DeleteRequest)$
  ELSE Nil

$WithDeleteResponse(m) \triangleq$
  IF $Assert(ValidDeleteResponse(m),$ "Invalid DeleteResponse")
  THEN $SetType(m,\ DeleteResponse)$
  ELSE Nil

$WithGetRequest(m) \triangleq$

```
    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

  WithListRequest(m) ≜
    IF Assert(ValidListRequest(m), "Invalid ListRequest")
     THEN SetType(m, ListRequest)
     ELSE Nil

  WithListResponse(m) ≜
    IF Assert(ValidListResponse(m), "Invalid ListResponse")
     THEN SetType(m, ListResponse)
     ELSE Nil

  WithWatchRequest(m) ≜
    IF Assert(ValidWatchRequest(m), "Invalid WatchRequest")
     THEN SetType(m, WatchRequest)
     ELSE Nil

  WithWatchResponse(m) ≜
    IF Assert(ValidWatchResponse(m), "Invalid WatchResponse")
     THEN SetType(m, WatchResponse)
     ELSE Nil
```

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

```
LOCAL Messages ≜ INSTANCE Messages WITH
  CreateRequest ← "CreateRequest",
  CreateResponse ← "CreateResponse",
  UpdateRequest ← "UpdateRequest",
  UpdateResponse ← "UpdateResponse",
  DeleteRequest ← "DeleteRequest",
  DeleteResponse ← "DeleteResponse",
  GetRequest ← "GetRequest",
  GetResponse ← "GetResponse",
  ListRequest  ← "ListRequest",
  ListResponse  ← "ListResponse",
  WatchRequest ← "WatchRequest",
  WatchResponse ← "WatchResponse"
```

MODULE *Client*

The *Client* module provides operators for managing and operating on *Topo* 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 *Topo* client.

$$CreateRequest(c,\ m) \triangleq \\ \quad \land\ gRPC!Client!Send(c,\ Messages!WithCreateRequest(m))$$

$$UpdateRequest(c,\ m) \triangleq \\ \quad \land\ gRPC!Client!Send(c,\ Messages!WithUpdateRequest(m))$$

$$DeleteRequest(c,\ m) \triangleq \\ \quad \land\ gRPC!Client!Send(c,\ Messages!WithDeleteRequest(m))$$

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

$$ListRequest(c,\ m) \triangleq \\ \quad \land\ gRPC!Client!Send(c,\ Messages!WithListRequest(m))$$

$$WatchRequest(c,\ m) \triangleq \\ \quad \land\ gRPC!Client!Send(c,\ Messages!WithWatchRequest(m))$$

Instantiate the *Topo*!*Client*!*Send* module

$Send \triangleq$ INSTANCE *Send*

MODULE *Receive*

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

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

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

$$DeleteResponse(c,\ h(\_,\ \_)) \triangleq \\ \quad gRPC!Client!Handle(c,\ \text{LAMBDA}\ x,\ m :$$

$\land$ *Messages*!*IsDeleteResponse*(m)
$\land$ *gRPC*!*Client*!*Receive*(c)
$\land$ $h(c,\ m)$))

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

ListResponse(c, $h(\_,\ \_)$) $\triangleq$
gRPC!*Client*!*Handle*(c, LAMBDA x, m :
$\land$ *Messages*!*IsListResponse*(m)
$\land$ *gRPC*!*Client*!*Receive*(c)
$\land$ $h(c,\ m)$))

WatchResponse(c, $h(\_,\ \_)$) $\triangleq$
gRPC!*Client*!*Handle*(c, LAMBDA x, m :
$\land$ *Messages*!*IsWatchResponse*(m)
$\land$ *gRPC*!*Client*!*Receive*(c)
$\land$ $h(c,\ m)$))

Instantiate the *Topo*!*Client*!*Receive* module
Receive $\triangleq$ INSTANCE *Receive*

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

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

Provides operators for the *Topo* client
Client $\triangleq$ INSTANCE *Client*

MODULE *Server*

The *Server* module provides operators for managing and operating on *Topo* 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 *Topo* server.

CreateResponse(c, m) $\triangleq$
$\land$ *gRPC*!*Server*!*Send*(c, *Messages*!*WithCreateResponse*(m))

UpdateResponse(c, m) $\triangleq$
$\land$ *gRPC*!*Server*!*Send*(c, *Messages*!*WithUpdateResponse*(m))

$DeleteResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithDeleteResponse(m))$

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

$ListResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithListResponse(m))$

$WatchResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Send(c,\ Messages!WithWatchResponse(m))$

Instantiate the *Topo*!*Server*!*Send* module
$Send \triangleq$ INSTANCE *Send*

MODULE *Reply*

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

$CreateResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithCreateResponse(m))$

$UpdateResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithUpdateResponse(m))$

$DeleteResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithDeleteResponse(m))$

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

$ListResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithListResponse(m))$

$WatchResponse(c,\ m) \triangleq$
$\quad \land\ gRPC!Server!Reply(c,\ Messages!WithWatchResponse(m))$

Instantiate the *Topo*!*Server*!*Reply* module
$Reply \triangleq$ INSTANCE *Reply*

MODULE *Receive*

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

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

```
        ∧ Messages!IsCreateRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))

  UpdateRequest(c, h(_, _)) ≜
    gRPC!Server!Handle(c, LAMBDA x, m :
        ∧ Messages!IsUpdateRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))

  DeleteRequest(c, h(_, _)) ≜
    gRPC!Server!Handle(c, LAMBDA x, m :
        ∧ Messages!IsDeleteRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))

  GetRequest(c, h(_, _)) ≜
    gRPC!Server!Handle(c, LAMBDA x, m :
        ∧ Messages!IsGetRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))

  ListRequest(c, h(_, _)) ≜
    gRPC!Server!Handle(c, LAMBDA x, m :
        ∧ Messages!IsListRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))

  WatchRequest(c, h(_, _)) ≜
    gRPC!Server!Handle(c, LAMBDA x, m :
        ∧ Messages!IsWatchRequest(m)
        ∧ gRPC!Server!Receive(c)
        ∧ h(c, m))
```

Instantiate the *Topo*!*Server*!*Receive* module

```
Handle ≜ INSTANCE Receive
```

Provides operators for the *Topo* server

```
Server ≜ INSTANCE Server
```

The set of all open *Topo* connections

```
Connections ≜ gRPC!Connections

Init ≜
```

$\land\ gRPC!Init$

$Next \triangleq$
$\land\ gRPC!Next$

* Modification History
* Last modified *Mon Sep* 13 19:22:31 *PDT* 2021 by *jordanhalterman*
* *Created Mon Sep* 13 16:24:05 *PDT* 2021 by *jordanhalterman*