MODULE *Topo*

The *Topo* 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*

Message type constants

CONSTANT
CreateRequestType,
CreateResponseType

CONSTANTS
UpdateRequestType,
UpdateResponseType

CONSTANTS
DeleteRequestType,
DeleteResponseType

CONSTANT
GetRequestType,
GetResponseType

CONSTANT
ListRequestType,
ListResponseType

CONSTANT
WatchRequestType,
WatchResponseType

LOCAL $messageTypes \triangleq$
$\{$*CreateRequestType*,
CreateResponseType,
UpdateRequestType,
UpdateResponseType,
DeleteRequestType,
DeleteResponseType,
GetRequestType,
GetResponseType,
ListRequestType,
ListResponseType,
WatchRequestType,
WatchResponseType$\}$

Message types should be defined as strings to simplify debugging

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

VARIABLE *conns*

LOCAL INSTANCE *API*

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*.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$CreateRequest(m) \triangleq$
  IF $Assert(ValidCreateRequest(m), \text{"Invalid CreateRequest"})$
  THEN $SetType(m, CreateRequestType)$
  ELSE Nil

$CreateResponse(m) \triangleq$
  IF $Assert(ValidCreateResponse(m), \text{"Invalid CreateResponse"})$
  THEN $SetType(m, CreateResponseType)$
  ELSE Nil

$UpdateRequest(m) \triangleq$
  IF $Assert(ValidUpdateRequest(m), \text{"Invalid UpdateRequest"})$
  THEN $SetType(m, UpdateRequestType)$
  ELSE Nil

$UpdateResponse(m) \triangleq$
  IF $Assert(ValidUpdateResponse(m), \text{"Invalid UpdateResponse"})$
  THEN $SetType(m, UpdateResponseType)$
  ELSE Nil

$DeleteRequest(m) \triangleq$
  IF $Assert(ValidDeleteRequest(m), \text{"Invalid DeleteRequest"})$
  THEN $SetType(m, DeleteRequestType)$
  ELSE Nil

$DeleteResponse(m) \triangleq$
  IF $Assert(ValidDeleteResponse(m), \text{"Invalid DeleteResponse"})$
  THEN $SetType(m, DeleteResponseType)$
  ELSE Nil

$GetRequest(m) \triangleq$
  IF $Assert(ValidGetRequest(m), \text{"Invalid GetRequest"})$
  THEN $SetType(m, GetRequestType)$
  ELSE Nil

$GetResponse(m) \triangleq$
IF $Assert(ValidGetResponse(m),$ "Invalid GetResponse")
THEN $SetType(m,\ GetResponseType)$
ELSE Nil

$ListRequest(m) \triangleq$
IF $Assert(ValidListRequest(m),$ "Invalid ListRequest")
THEN $SetType(m,\ ListRequestType)$
ELSE Nil

$ListResponse(m) \triangleq$
IF $Assert(ValidListResponse(m),$ "Invalid ListResponse")
THEN $SetType(m,\ ListResponseType)$
ELSE Nil

$WatchRequest(m) \triangleq$
IF $Assert(ValidWatchRequest(m),$ "Invalid WatchRequest")
THEN $SetType(m,\ WatchRequestType)$
ELSE Nil

$WatchResponse(m) \triangleq$
IF $Assert(ValidWatchResponse(m),$ "Invalid WatchResponse")
THEN $SetType(m,\ WatchResponseType)$
ELSE Nil

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

LOCAL $Messages \triangleq$ INSTANCE $Messages$

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 *Requests*

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

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

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

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

$GetRequest(c,\ m) \triangleq$

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

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

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

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

MODULE *Responses*

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

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

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

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

$WatchResponse(c,\ h(\_,\ \_))\ \triangleq$

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

Instantiate the $Topo!Client!Receive$ module
$Receive \triangleq \text{INSTANCE } Responses$

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

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

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

MODULE *Server*

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

MODULE *Responses*

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

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

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

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

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

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

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

Instantiate the $Topo!Server!Send$ module
$Send \triangleq \text{INSTANCE } Responses$

MODULE *Requests*

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

$$
\begin{aligned}
&\mathit{CreateRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsCreateRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{UpdateRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsUpdateRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{DeleteRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsDeleteRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{GetRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsGetRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ListRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsListRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{WatchRequest}(c,\ h(\_,\ \_)) \triangleq \\
&\quad \mathit{gRPC}!\mathit{Server}!\mathit{Handle}(c,\ \text{LAMBDA}\ x,\ m : \\
&\qquad \land\ \mathit{Messages}!\mathit{IsWatchRequest}(m) \\
&\qquad \land\ \mathit{gRPC}!\mathit{Server}!\mathit{Receive}(c) \\
&\qquad \land\ h(c,\ m))
\end{aligned}
$$

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

$$\mathit{Receive} \triangleq \text{INSTANCE}\ \mathit{Requests}$$

Provides operators for the *Topo* server

$Server \triangleq \text{INSTANCE } Server$

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

* Modification History
* Last modified *Mon Sep* 13 15:16:04 *PDT* 2021 by *jordanhalterman*
* *Created Mon Sep* 13 15:07:05 *PDT* 2021 by *jordanhalterman*