
MODULE *E2TService*

The *E2TService* module provides a formal specification of the *E2T* service. The spec defines the client and server interfaces for *E2T* and provides helpers for managing and operating on connections.

CONSTANT *Nil*

VARIABLE *conns*

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

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 *E2T*.

Message type constants

CONSTANT

$SubscribeRequest$,
 $SubscribeResponse$

CONSTANTS

$UnsubscribeRequest$,
 $UnsubscribeResponse$

CONSTANTS

$ControlRequest$,
 $ControlResponse$

LOCAL $messageTypes \triangleq$

 { $SubscribeRequest$,
 $SubscribeResponse$,
 $UnsubscribeRequest$,
 $UnsubscribeResponse$,
 $ControlRequest$,
 $ControlResponse$ }

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 *E2T* message types on the network.

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

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

$IsUnsubscribeRequest(m) \triangleq m.type = UnsubscribeRequest$
 $IsUnsubscribeResponse(m) \triangleq m.type = UnsubscribeResponse$
 $IsControlRequest(m) \triangleq m.type = ControlRequest$
 $IsControlResponse(m) \triangleq m.type = ControlResponse$

This section defines predicates for validating *E2T* message contents. The predicates provide precise documentation on the *E2T* message format and are used within the spec to verify that steps adhere to the *E2T* protocol specification.

LOCAL $ValidSubscribeRequest(m) \triangleq \text{TRUE}$
 LOCAL $ValidSubscribeResponse(m) \triangleq \text{TRUE}$
 LOCAL $ValidUnsubscribeRequest(m) \triangleq \text{TRUE}$
 LOCAL $ValidUnsubscribeResponse(m) \triangleq \text{TRUE}$
 LOCAL $ValidControlRequest(m) \triangleq \text{TRUE}$
 LOCAL $ValidControlResponse(m) \triangleq \text{TRUE}$

This section defines operators for constructing *E2T* messages.

LOCAL $SetType(m, t) \triangleq [m \text{ EXCEPT } !.type = t]$
 $WithSubscribeRequest(m) \triangleq$
 IF $Assert(ValidSubscribeRequest(m), \text{"Invalid SubscribeRequest"})$
 THEN $SetType(m, SubscribeRequest)$
 ELSE Nil
 $WithSubscribeResponse(m) \triangleq$
 IF $Assert(ValidSubscribeResponse(m), \text{"Invalid SubscribeResponse"})$
 THEN $SetType(m, SubscribeResponse)$
 ELSE Nil
 $WithUnsubscribeRequest(m) \triangleq$
 IF $Assert(ValidUnsubscribeRequest(m), \text{"Invalid UnsubscribeRequest"})$
 THEN $SetType(m, UnsubscribeRequest)$
 ELSE Nil
 $WithUnsubscribeResponse(m) \triangleq$
 IF $Assert(ValidUnsubscribeResponse(m), \text{"Invalid UnsubscribeResponse"})$
 THEN $SetType(m, UnsubscribeResponse)$
 ELSE Nil
 $WithControlRequest(m) \triangleq$

```

IF Assert(ValidControlRequest(m), "Invalid ControlRequest")
THEN SetType(m, ControlRequest)
ELSE Nil

```

```

WithControlResponse(m)  $\triangleq$ 
IF Assert(ValidControlResponse(m), "Invalid ControlResponse")
THEN SetType(m, ControlResponse)
ELSE Nil

```

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

```

LOCAL Messages  $\triangleq$  INSTANCE Messages WITH
  SubscribeRequest  $\leftarrow$  "SubscribeRequest",
  SubscribeResponse  $\leftarrow$  "SubscribeResponse",
  UnsubscribeRequest  $\leftarrow$  "UnsubscribeRequest",
  UnsubscribeResponse  $\leftarrow$  "UnsubscribeResponse",
  ControlRequest  $\leftarrow$  "ControlRequest",
  ControlResponse  $\leftarrow$  "ControlResponse"

```

MODULE *Client*

The *Client* module provides operators for managing and operating on *E2T* 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 *E2T* client.

```

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

UnsubscribeRequest(c, m)  $\triangleq$ 
   $\wedge$  gRPC!Client!Send(c, Messages!WithUnsubscribeRequest(m))

ControlRequest(c, m)  $\triangleq$ 
   $\wedge$  gRPC!Client!Send(c, Messages!WithControlRequest(m))

```

Instantiate the *E2T!Client!Requests* module

```

Send  $\triangleq$  INSTANCE Requests

```

MODULE *Responses*

This module provides predicates for the types of messages that can be received by an *E2T* client.

```

SubscribeResponse(c, h(-))  $\triangleq$ 
  gRPC!Client!Handle(c, LAMBDA x, m :

```

$$\begin{aligned} & \wedge \text{Messages!IsSubscribeResponse}(m) \\ & \wedge \text{gRPC!Client!Receive}(c) \\ & \wedge h(m)) \end{aligned}$$

$$\begin{aligned} \text{UnsubscribeResponse}(c, h(-)) & \triangleq \\ & \text{gRPC!Client!Handle}(c, \text{LAMBDA } x, m : \\ & \quad \wedge \text{Messages!IsUnsubscribeResponse}(m) \\ & \quad \wedge \text{gRPC!Client!Receive}(c) \\ & \quad \wedge h(m)) \end{aligned}$$

$$\begin{aligned} \text{ControlResponse}(c, h(-)) & \triangleq \\ & \text{gRPC!Client!Handle}(c, \text{LAMBDA } x, m : \\ & \quad \wedge \text{Messages!IsControlResponse}(m) \\ & \quad \wedge \text{gRPC!Client!Receive}(c) \\ & \quad \wedge h(m)) \end{aligned}$$

Instantiate the *E2T!Client!Responses* module
 $\text{Handle} \triangleq \text{INSTANCE Responses}$

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

$$\text{Disconnect}(c) \triangleq \text{gRPC!Client!Disconnect}(c)$$

Provides operators for the *E2T* client
 $\text{Client} \triangleq \text{INSTANCE Client}$

MODULE *Server*

The *Server* module provides operators for managing and operating on *E2T* 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 *E2T* server.

$$\begin{aligned} \text{SubscribeResponse}(c, m) & \triangleq \\ & \wedge \text{gRPC!Server!Reply}(c, \text{Messages!WithSubscribeResponse}(m)) \end{aligned}$$

$$\begin{aligned} \text{UnsubscribeResponse}(c, m) & \triangleq \\ & \wedge \text{gRPC!Server!Reply}(c, \text{Messages!WithUnsubscribeResponse}(m)) \end{aligned}$$

$$\begin{aligned} \text{ControlResponse}(c, m) & \triangleq \\ & \wedge \text{gRPC!Server!Reply}(c, \text{Messages!WithControlResponse}(m)) \end{aligned}$$

Instantiate the *E2T!Server!Responses* module

$Send \triangleq \text{INSTANCE } Responses$

MODULE *Requests*

This module provides predicates for the types of messages that can be received by an *E2T* server.

$SubscribeRequest(c, h(-)) \triangleq$
 $gRPC!Server!Handle(c, \text{LAMBDA } x, m :$
 $\quad \wedge Messages!IsSubscribeRequest(m)$
 $\quad \wedge gRPC!Server!Receive(c)$
 $\quad \wedge h(m))$

$UnsubscribeRequest(c, h(-)) \triangleq$
 $gRPC!Server!Handle(c, \text{LAMBDA } x, m :$
 $\quad \wedge Messages!IsUnsubscribeRequest(m)$
 $\quad \wedge gRPC!Server!Receive(c)$
 $\quad \wedge h(m))$

$ControlRequest(c, h(-)) \triangleq$
 $gRPC!Server!Handle(c, \text{LAMBDA } x, m :$
 $\quad \wedge Messages!IsControlRequest(m)$
 $\quad \wedge gRPC!Server!Receive(c)$
 $\quad \wedge h(m))$

Instantiate the *E2T!Server!Requests* module
 $Handle \triangleq \text{INSTANCE } Requests$

Provides operators for the *E2T* server
 $Server \triangleq \text{INSTANCE } Server$

The set of all open *E2T* connections
 $Connections \triangleq gRPC!Connections$

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

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

\ * Modification History
 \ * Last modified *Mon Sep 13 18:18:42 PDT 2021* by *jordanhalterman*
 \ * Created *Mon Sep 13 16:23:16 PDT 2021* by *jordanhalterman*