MODULE $E2T$

The $E2AP$ 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
  $OK \leftarrow$ "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 $E2T$.

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 $\forall m \in messageTypes : m \in$ 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) ≜ TRUE

LOCAL ValidSubscribeResponse(m) ≜ TRUE

LOCAL ValidUnsubscribeRequest(m) ≜ TRUE

LOCAL ValidUnsubscribeResponse(m) ≜ TRUE

LOCAL ValidControlRequest(m) ≜ TRUE

LOCAL ValidControlResponse(m) ≜ TRUE
```

This section defines operators for constructing $E2T$ messages.

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

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

WithSubscribeResponse(m) ≜
   IF Assert(ValidSubscribeResponse(m), "Invalid SubscribeResponse")
    THEN SetType(m, SubscribeResponse)
    ELSE Nil

WithUnsubscribeRequest(m) ≜
   IF Assert(ValidUnsubscribeRequest(m), "Invalid UnsubscribeRequest")
    THEN SetType(m, UnsubscribeRequest)
    ELSE Nil

WithUnsubscribeResponse(m) ≜
   IF Assert(ValidUnsubscribeResponse(m), "Invalid UnsubscribeResponse")
    THEN SetType(m, UnsubscribeResponse)
    ELSE Nil

WithControlRequest(m) ≜
```

```
    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.

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

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

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

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 $x, m$ :
            ∧ Messages!IsSubscribeResponse(m)
            ∧ gRPC!Client!Receive(c)
            ∧ h(c, m))

    UnsubscribeResponse(c, h(_, _)) ≜
        gRPC!Client!Handle(c, LAMBDA x, m :
            ∧ Messages!IsUnsubscribeResponse(m)
            ∧ gRPC!Client!Receive(c)
            ∧ h(c, m))

    ControlResponse(c, h(_, _)) ≜
        gRPC!Client!Handle(c, LAMBDA x, m :
            ∧ Messages!IsControlResponse(m)
            ∧ gRPC!Client!Receive(c)
            ∧ h(c, m))
```

Instantiate the *E2T*!*Client*!*Responses* module

```
  Receive ≜ INSTANCE Responses

  Connect(s, d) ≜ gRPC!Client!Connect(s, d)

  Disconnect(c) ≜ gRPC!Client!Disconnect(c)
```

Provides operators for the *E2T* client

```
Client ≜ 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.

```
    SubscribeResponse(c, m) ≜
        ∧ gRPC!Server!Reply(c, Messages!WithSubscribeResponse(m))

    UnsubscribeResponse(c, m) ≜
        ∧ gRPC!Server!Reply(c, Messages!WithUnsubscribeResponse(m))

    ControlResponse(c, m) ≜
        ∧ gRPC!Server!Reply(c, Messages!WithControlResponse(m))
```

Instantiate the *E2T*!*Server*!*Responses* module

```
Send ≜ INSTANCE Responses

    ──────────────── MODULE Requests ────────────────

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

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

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

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

    ─────────────────────────────────────────────────

  Instantiate the E2T!Server!Requests module
Receive ≜ INSTANCE Requests

 ──────────────────────────────────────────────────

 Provides operators for the E2T server
Server ≜ INSTANCE Server

 The set of all open E2T connections
Connections ≜ gRPC!Connections

Init ≜
   ∧ gRPC!Init

Next ≜
   ∧ gRPC!Next

────────────────────────────────────────────────────
\* Modification History
\* Last modified Mon Sep 13 15:34:44 PDT 2021 by jordanhalterman
\* Created Mon Sep 13 14:04:44 PDT 2021 by jordanhalterman
```