Reliable Message Transport Protocol API

Version 1.10

Allen Baker

Hewlett-Packard Company
INSO R&D Lab

May 26, 2014

1. Revisions

1.1 Version 1.10 (25oct96)

- rmtpInitialize() now returns TcCode instead of void.

1.2 Version 1.05 (26aug96)

- changed rmtpGetNextEvent() to take a pointer to a pointer to a Trply instead of a pointer to a Trply.

1.3 Version 1.04 (15aug96)

- Added prmLocalAddrPtr to rmtpCall() parameter list.
- Specified exactly how Taddr parameters are used.
- Defined how port fields in Taddrs are used.

1.4 Version 1.03 (17jul96)

- Cleared up confusion about state of Tbcb* parameters in rmtpSend() and rmtpReceive().

1.5 Version 1.02 (28jun96)

- Added ETcCodeInvalidConn as a possible return code for all functions that take a connection handle as a parameter.

1.6 Version 1.01 (14jun96)

- removed in and out parameter class specifiers.

1.7 Version 1.00

- Initial.

2. Patents

The protocol described in this document is protected under the following international patents which are owned by the Hewlett-Packard Company:

1. DE69802792D1
2. DE69802792T2
3. DE69835314D1
4. DE69835314T2
5. EP0872991A2
6. EP0872991A3
7. EP0872991B1
8. EP0886411A2
9. EP0886411A3
10. EP0976058A1
11. EP0976058B1
12. US6202096 B1
13. US6502000
14. US6721286
15. WO1998047076A1

3. Introduction

An External Reference Specification (ERS) specifies its target object's look, feel, and functionality from an external perspective. The external perspective is a view of the target object from the outside and does not include requirements specifications, the target object's internal architectural and design details, or the target object's architectural relationships with other subsystems and modules (the target object's relationships to other subsystems and modules is described in the Architecture Specification). The external perspective includes all of the target object's hardware and software interfaces, protocols, performance characteristics, functionality, and operational procedures. Interfaces are inputs and outputs like those to and from users and other systems, subsystems and modules. An ERS describes its target object's interfaces and functionalities as a whole, without describing how those interfaces and functionalities are allocated to subsystems and modules within the target object.

The target object for this ERS is the Reliable Message Transport Protocol API.

4. Application Program Interface (API)

```
/*
$Header: tops_rmtp.h,v 1.7 96/10/15 15:34:33 jmeltz Exp $
*/

#if !defined(_IGNORE_RMTP_H)
   #define _IGNORE_RMTP_H


   /* ================================================================ */
   /* ||||||||| (c) Copyright Hewlett-Packard Company 1996. |||||||||| */
   /* |||||||||             All Rights Reserved.            |||||||||| */
   /* ================================================================ */


   /* ================================================================ */
   /* ||||||||||||||||||||||||| Description ||||||||||||||||||||||||| */
   /* ================================================================ */


   /*
   ================================================

   - TAB size is 3.

   - Version information is maintained in RMTPERS.Doc

   This file defines the RMTP API.  The API consists of all the constant,
   type, function and event (events are called replies in this API)
   definitions that the OSP needs to access the RMTP transport service
   provider within the local host.

   OSP does not make these calls directly.  Instead, it calls TIAPI which
   in most cases simply passes the call through to RMTP.
   ------------------------------------------------ */


   /* ================================================================ */
   /* |||||||||||||||||||||| Included Files |||||||||||||||||||||||| */
   /* ================================================================ */


   #include "appl_tiapi.h"



   /* ================================================================ */
   /* |||||||||||||||||| Prototypes, Replies and Events ||||||||||||||| */
   /* ================================================================ */


   /*
   ================================================================
   [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
   ================================================================

   Name:
      rmtpInitialize()
```

```
Description:
   Initialize or startup RMTP.

   This must be called before any other RMTP functions are called.
   It can be called only one time, when RMTP is started.

Parameters:
   none

Returns:
   ETcCodeSuccess
      if initialization completed successfully.
   ETcCodeFailure
      if RMTP cannot initialize properly.  If this is returned, RMTP
      will not function.

-----------------------------------------------------------------
*/
TcCode   rmtpInitialize
   (
   void
   );



/*
=================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
=================================================================

Name:
   rmtpShutdown()

Description:
   Shutdown RMTP.

   When this call is made, all pending requests are canceled by
   RMTP.  No replies are generated for the canceled requests.  The
   OSP layer should assume that all of its outstanding RMTP requests
   are canceled when this function returns.  RMTP will return no more
   replies through rmtpGetNextEvent().

   This should be called only one time, when RMTP is terminated.
   After this call RMTP will not function again until a
   rmtpInitialize() call is completed.

Parameters:
   none

Returns:
   none

-----------------------------------------------------------------
*/
void  rmtpShutdown
   (
   void
   );
```

```
/*
====================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
====================================================================

Name:
   rmtpCancelRequest()

Description:
   Cancel a previously issued request.

   RMTP will cancel all pending requests that have the specified
   prmRqstHdl value.

Parameters:
   prmRqstHdl:
      The request handle that was passed into RMTP by OSP in the
      function call that made the request.

Returns:
   ETcCodeSuccess:
      If the request was successfully canceled.
   ETcRequestNotFound
      RMTP cannot find the specified request.
   ETcCodeFailure:
      The request was found but cannot be canceled.  This code may
      also be returned by a RMTP implementation that does not
      support canceling the previously issued request.  (Some
      implementations may only cancel rmtpListen() requests.)

--------------------------------------------------------------------
*/
TcCode   rmtpCancelRequest
   (
   TrqstHdl prmRqstHdl
   );



/*
====================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
====================================================================

Name:
   rmtpRun()

Description:
   Accept the thread of control from the caller.

   TOPS is single threaded for portability reasons.  One way RMTP
   gets hold of the thread is through this call.  When RMTP gets the
   thread it can use it for whatever processing it needs to do.
   Typically, RMTP would respond to this call by checking for events
   from the underlying transports.

   When RMTP has nothing to do, it returns the thread by returning
   from this function.

Parameters:
   none
```

```
Returns:
   none

-----------------------------------------------------------------
*/
void  rmtpRun
   (
   void
   );



/*
====================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
====================================================================

Name:
   rmtpCall()

Description:
   Attempt to open a connection to another device.

   The response to this call is a reply that is defined in TIAPI.H:
      TrplyActiveConnect

   - This function may send or receive data over the network and so -
   - is implemented as an asynchronous (non-blocking) call.  The    -
   - results of its action are returned as a corresponding reply.   -

   A platform that is multi-homed is one that has more than one
   local address within the same domain.  For example, a LaserJet with
   2 JetDirect cards may have 2 IP addresses.  In some cases it may be
   possible to connect to a foreign peer through one of those
   local addresses but not the other.  If prmLocalAddrPtr is NULL,
   rmtpCall() will select the local address through which it can
   establish a connection to the foreign peer -- if any.

Parameters:
   prmRqstHdl:
      An OSP generated request handle that RMTP will include in its
      reply to OSP that corresponds to this call.  OSP can use the
      handle to match-up its calls to asynchronous functions with the
      replies it gets back for those functions.
   prmLocalAddrPtr:
      A pointer to a RMTP address structure containing the address
      of the local host through which RMTP should attempt to
      establish a connection.  If prmLocalAddressPtr is NULL,
      RMTP will select the local address through which it will
      attempt to establish the connection (same behavior as using
      INET_ANY in BerkeleySockets).  If it is not NULL, RMTP will
      use the local address it points to when attempting to establish
      the connection.
         - If the port field (or socket field in the case of SIPX
           domain) in the address is zero, RMTP will call through
           its well-known RMTP port.  If prmLocalAddrPtr is
           NULL, the port field is treated as if it were set to
           zero.
         - Otherwise, RMTP will open the specified port and call
           through it.
   prmForeignAddrPtr:
      A pointer to a RMTP address structure containing the address
```

```
      of the foreign peer with which RMTP should attempt to
      establish a connection.
         - If the port field (or socket field in the case of SIPX
           domain) in the address is zero, RMTP will call to
           the well-known RMTP port.
         - Otherwise, RMTP will call to the specified port.

Returns:
   ETcCodeSuccess:
      If the request was successfully enqueued by RMTP.  RMTP will
      act upon the request and generate an asynchronous reply to it.
   ETcCodeFailure:
      RMTP could not enqueue the request.  This may happen if RMTP
      is out of queue resources.

Semantics:

-----------------------------------------------------------------
*/
TcCode   rmtpCall
   (
   TrqstHdl prmRqstHdl,
   Taddr*   prmLocalAddrPtr,
   Taddr*   prmForeignAddrPtr
   );



/*
=================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
=================================================================

Name:
   rmtpListen()

Description:
   Accept connection requests from other devices.

   The response to this call is a reply that is defined in TIAPI.H:
      TrplyPassiveConnect

   - This function may send or receive data over the network and so -
   - is implemented as an asynchronous (non-blocking) call.  The    -
   - results of its action are returned as a corresponding reply.   -

Parameters:
   prmRqstHdl:
      An OSP generated request handle that RMTP will include in its
      reply to OSP that corresponds to this call.  OSP can use the
      handle to match-up its calls to asynchronous functions with the
      replies it gets back for those functions.
   prmLocalAddrPtr:
      A pointer to a RMTP address structure containing the address
      of the local host over which RMTP should listen for connection
      attempts.
         - If the port field (or socket field in the case of SIPX
           domain) in the address is zero, RMTP will listen
           through its well-known RMTP port.
         - Otherwise, RMTP will open the specified port and
           listen through it.
```

```
Returns:
   ETcCodeSuccess:
      If the request was successfully enqueued by RMTP.  RMTP will
      act upon the request and generate asynchronous replies to it.
   ETcCodeFailure:
      RMTP could not enqueue the request.  This may happen if RMTP
      is out of queue resources.

Semantics:

-----------------------------------------------------------------
*/
TcCode   rmtpListen
   (
   TrqstHdl prmRqstHdl,
   Taddr*   prmLocalAddrPtr
   );



/*
=================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
=================================================================

Name:
   rmtpHangUp()

Description:
   Close a connection to another device.

   The response to this call is a reply that is defined in TIAPI.H:
      TrplyDisconnect

   * This request cannot be canceled. *

   - This function may send or receive data over the network and so -
   - is implemented as an asynchronous (non-blocking) call.  The    -
   - results of its action are returned as a corresponding reply.   -

Parameters:
   prmRqstHdl:
      An OSP generated request handle that RMTP will include in its
      reply to OSP that corresponds to this call.  OSP can use the
      handle to match-up its calls to asynchronous functions with the
      replies it gets back for those functions.
   prmConnHdl:
      Identifier, previously generated and returned by RMTP, that
      tells RMTP the connection upon which it should perform this
      operation.

Returns:
   ETcCodeSuccess:
      If the request was successfully enqueued by RMTP.  RMTP will
      act upon the request and generate an asynchronous reply to it.
   ETcCodeFailure:
      RMTP could not enqueue the request.  This may happen if RMTP
      is out of queue resources.
   ETcCodeInvalidConn:
      The specified connection handle is invalid.

Semantics:
```

```
-------------------------------------------------------------
*/
TcCode   rmtpHangUp
   (
   TrqstHdl prmRqstHdl,
   TconnHdl prmConnHdl
   );



/*
=============================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
=============================================================

Name:
   rmtpSend()

Description:
   Send a buffer of data through a connection to another device.

   The response to this call is a reply that is defined in TIAPI.H:
      TrplySend

   - This function may send or receive data over the network and so -
   - is implemented as an asynchronous (non-blocking) call.  The    -
   - results of its action are returned as a corresponding reply.   -

Parameters:
   prmRqstHdl:
      An OSP generated request handle that RMTP will include in its
      reply to OSP that corresponds to this call.  OSP can use the
      handle to match-up its calls to asynchronous functions with the
      replies it gets back for those functions.
   prmConnHdl:
      Identifier, previously generated and returned by RMTP, that
      tells RMTP the connection upon which it should perform this
      operation.
   prmDataPtr:
      A pointer to a buffer control block that rmtpSend()
      will use to send data.  rmtpSend() will assume
      that the payload portion of the Tbcb is large enough
      to send prmMaxBytes of data from.  RMTP will
      simply send data from the buffer at the address returned
      by calling BufDataAddr() on the Tbcb.
   prmBytesToSend:
      The length of the data pointed to by prmDataPtr.

Returns:
   ETcCodeSuccess:
      If the request was successfully enqueued by RMTP.  RMTP will
      act upon the request and generate an asynchronous reply to it.
   ETcCodeFailure:
      RMTP could not enqueue the request.  This may happen if RMTP
      is out of queue resources.
   ETcCodeInvalidConn:
      The specified connection handle is invalid.

Semantics:

-------------------------------------------------------------
```

```
*/
TcCode   rmtpSend
   (
   TrqstHdl prmRqstHdl,
   TconnHdl prmConnHdl,
   Tbcb*    prmDataPtr,
   Ts32     prmBytesToSend
   );



/*
====================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
====================================================================

Name:
   rmtpReceive()

Description:
   Accept a buffer of data through a connection to another device.
   This function passes buffers into RMTP that RMTP can use for
   receiving data over a connection.

   The response to this call is a reply that is defined in TIAPI.H:
      TrplyReceive

   - This function may send or receive data over the network and so -
   - is implemented as an asynchronous (non-blocking) call.  The    -
   - results of its action are returned as a corresponding reply.   -

Parameters:
   prmRqstHdl:
      An OSP generated request handle that RMTP will include in its
      reply to OSP that corresponds to this call.  OSP can use the
      handle to match-up its calls to asynchronous functions with the
      replies it gets back for those functions.
   prmConnHdl:
      Identifier, previously generated and returned by RMTP, that
      tells RMTP the connection upon which it should perform this
      operation.
   prmMaxBytes:
      The maximum number of bytes that can be written into
      *prmDataPtr.
   prmDataPtr:
      A pointer to a buffer control block that rmtpReceive()
      will use to receive data.  rmtpReceive() will assume
      that the payload portion of the Tbcb is large enough
      to receive prmMaxBytes of data.  Prior to calling
      rmtpReceive(), OSP will have called PrependBuf() to
      make room in the Tbcb for prmMaxBytes of data.  RMTP will
      simply receive data into the buffer at the address returned
      by calling BufDataAddr() on the Tbcb.

Returns:
   ETcCodeSuccess:
      If the request was successfully enqueued by RMTP.  RMTP will
      act upon the request and generate an asynchronous reply to it.
   ETcCodeFailure:
      RMTP could not enqueue the request.  This may happen if RMTP
      is out of queue resources.
   ETcCodeInvalidConn:
```

The specified connection handle is invalid.

Semantics:

```
------------------------------------------------------------------
*/
TcCode   rmtpReceive
   (
   TrqstHdl prmRqstHdl,
   TconnHdl prmConnHdl,
   Ts32     prmMaxBytes,
   Tbcb*    prmDataPtr
   );



/*
====================================================================
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
====================================================================
```

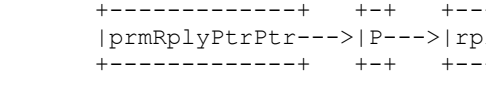
Name:
rmtpGetNextEvent()

Description:
Pass the next RMTP reply up to the TIAPI.

TIAPI will call this function to retrieve replies from the RMTP. RMTP will store the replies that it generates on an internal queue and pass the next one in order up to the TIAPI when this function is called.

Parameters:
prmRplyPtrPtr:
A pointer to a pointer to a reply structure. The reply structure pointer pointed to by prmRplyPtrPtr will be loaded with the address of the next reply from RMTP. Example:

RMTP owns a Trply named "rplyStruct" which it has filled in and is waiting to give to TIAPI. TIAPI owns a pointer to a Trply named "P" that is not pointing to anything. TIAPI passes a pointer to P to RMTP with the call rmtpGetNextEvent(&P). RMTP loads P with a pointer to rplyStruct and returns.




```
{
Trply*  P;

rmtpGetNextEvent(&P);

// P now contains the address of (pointer to)
// rplyStruct.
}
```

Returns:
ETcCodeSuccess:
If *prmRplyPtr contains a valid reply upon return.
ETcCodeFailure:

```
         Otherwise.  OSP can assumed this to mean that there is no reply
         to retrieve at the time of the call.

   ------------------------------------------------------------------
   */
   TcCode   rmtpGetNextEvent
      (
      Trply**  prmRplyPtrPtr
      );



   /*
   ====================================================================
   [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
   ====================================================================

   Name:
      rmtpNetworkEvent()

   Description:
      Process a network event.

      This function is called by TIAPI when there is a network event
      that RMTP needs to know about.  Examples are when a local network
      address becomes available or unavailable.

   Parameters:
      prmEventPtr:
         A pointer to a device defined event message.

   Returns:
      nothing.

   ------------------------------------------------------------------
   */
   void  rmtpNetworkEvent
      (
      void* prmEventPtr
      );


#endif  /* #if !defined(_IGNORE_RMTP_H) */
```