Tapwave, Inc. Proprietary

Tapwave®

Tapwave Multiplayer API Reference

Current Version	Date	Author
REV 0.3	02/27/2004	EM

Revision History

Version	Date	Description	Author
REV 0.1	12/14/2003	INITIAL REVISION	EM
REV 0.2	02/08/2004	UPDATE FOR BETA SDK RELEASE	EM
REV 0.3	02/27/2004	UPDATE FOR FINAL RELEASE	EM

Background

The purpose of the multiplayer API is to allow applications to start multiplayer games in a way that is easier for the gamer. It is not just an API, but a bluetooth service that runs on the device to handle incoming bluetooth multiplayer gaming invitations.

Using just the BtLib APIs, the app can use the Host/Join model to start a game where both players must be running the game and find their respective menus to properly initiate a game. With the TwMp APIs the game just needs to be present on the device or an inserted SD card and it will be launched by the TwMp service in order to start the game. If someone wants to host a multiplayer game with TwMp they just start the game, find devices to invite, and those devices are asked to join the game. The standard PalmOS BtLib APIs do not allow inviting someone to play a game that is not currently running the game (Invitees would just see a connecting dialog, with no explanation). Refer to the TwMpTest sample application for an example of an app that uses the TwMp APIs.

Multiplayer Game Process

The host enters the game and decides to host the game. All players press their bluetooth button to make the devices discoverable. The host finds the players he/she wants to invite to join the game via bluetooth inquiry and name discovery. The selected devices receive an invitation to join the game that they can either accept or reject. If the invitation is accepted, the game is launched with a PalmOS feature set so that the game can detect that it was launched for a multiplayer game.

Constants

```
#define kTwMpName                       "Tapwave Multiplayer Library"
#define sysFileCTwMp                    'twMU'
#define twMpFtrCreator                  sysFileCTwMp

/*
 * Error Codes
 */
#define twMpErrNoError                  (0)
#define twMpErrError                    (twMpErrorBase | 0xFF)
#define twMpErrNotOpen                  (twMpErrorBase | 0x01)
#define twMpErrNotHost                  (twMpErrorBase | 0x02)
#define twMpErrInvitationRejected       (twMpErrorBase | 0x03)
#define twMpErrHost                     (twMpErrorBase | 0x04)
#define twMpErrOutOfMemory              (twMpErrorBase | 0x05)
#define twMpErrNotEnough                (twMpErrorBase | 0x06)
#define twMpErrInProgress               (twMpErrorBase | 0x07)
#define twMpErrParamError               (twMpErrorBase | 0x08)
#define twMpErrTooMany                  (twMpErrorBase | 0x09)
#define twMpErrPending                  (twMpErrorBase | 0x0A)
#define twMpErrNoAccepted               (twMpErrorBase | 0x0B)
#define twMpErrLeftGame                 (twMpErrorBase | 0x0C)
#define twMpErrDisconnected             (twMpErrorBase | 0x0D)
#define twMpErrNotHostOrGuest           (twMpErrorBase | 0x0E)
#define twMpErrAlreadyOpen              (twMpErrorBase | 0x0F)
#define twMpErrNoConnection             (twMpErrorBase | 0x10)
#define twMpErrBadObjectVersion         (twMpErrorBase | 0x11)
#define twMpErrUIBusy                   (twMpErrorBase | 0x12)
```

```
#define twMpErrAlreadyConnected                 (twMpErrorBase | 0x13)
#define twMpErrNotFound                         (twMpErrorBase | 0x14)
#define twMpErrDisinvited                       (twMpErrorBase | 0x15)
#define twMpErrNotLaunched                      (twMpErrorBase | 0x16)

/*
 * Feature Numbers
 */

//
// Feature exists if the app was launched by the multiplayer library.  It must be checked by an application
// on application launch with sysAppLaunchCmdNormalLaunch to see if the app was launched by the
// multiplayer
// library.  The feature stores a pointer to a TwMpLaunchParamsType structure (68K Big Endian byte
// order)
//

#define twMpFtrNumMultiplayerLaunch             1

// This feature stores the version of the library.
#define twMpFtrNumVersion                       2


/*
 * Launch Codes
 */

//
// The OS does not allow reservation of app launch codes.
// So in order to avoid conflict with any potential custom launch codes you may want to add to your
// application we set our base for the custom launch codes you need to support if you use this API
// to sysAppLaunchCmdCustomBase + 0x6FFF, or 0xEFFF. The app will be sublaunched with this code to
// allow replacement of the Invitation dialog.
//
#define twMpAppLaunchCmdCustomBase              (sysAppLaunchCmdCustomBase + 0x6FFF)

// Allow the app to replace the default UI for asking the user
// if he/she wants to join the game.
#define twMpAppLaunchCmdAskUser                         (twMpAppLaunchCmdCustomBase)

/*
 * Misc. Constants
 */

// Results returned from PilotMain when sublaunched with twMpAppLaunchCmdAskUser
// to override the default invitation UI.
#define twMpResultAsk       (0) // Default invitation dialog will be displayed.
#define twMpResultAccept    (1) // Invitation Accepted (app will be launched by TwMp)
#define twMpResultDecline   (2) // Invitation Declined

// You can have 8 devices in a piconet which means 7 other players
#define kTwMpMaxOtherPlayers  (7)
```

Data Types and Structures

```
// Possible player states

// This player is hosting the game.
#define  twMpPlayerHosting       1

// Waiting for bluetooth connection to get in place
#define twMpPlayerWaiting       2

// Connected to device looking for Multiplayer service
#define  twMpPlayerAclConnected  3

// Connected to service.  Player being asked to join the game.
#define twMpPlayerConnected     4

// Player has accepted the game invitation
#define  twMpPlayerAccepted      5

// Player has declined the game invitation
#define  twMpPlayerDeclined      6

// Conection to bluetooth device failed
#define  twMpPlayerFailed        7

typedef UInt32 TwMpPlayerStateEnum;

// Update the connection status of a player
#define  twMpEventPlayerState        1

// Game has been canceled.  Or if you are a guest, you have been
// removed from the game.
// IMPORTANT: the btlib is left open by the multiplayer library
// so that the ACL links can be left in place.  This means you do
// not have to call BtLibOpen to use the BtLib APIs.  A call to TwMpClose
// will close the BtLib for the multiplayer library.  Since the BtLib
// keeps an open count you also need to close the library for every time
// that it is opened independently by the application as well.
#define  twMpEventGameCanceled       2

// For non-hosts, Initiate listener sockets and advertise them
// so the host can connect to them for the game when it receives
// the twMpEventHostStartGameEvent.
// IMPORTANT: the btlib is left open by the multiplayer library
// so that the ACL links can be left in place.  This means you do
// not have to call BtLibOpen to use the BtLib APIs.  A call to TwMpClose
// will close the BtLib for the multiplayer library.  Since the BtLib
// keeps an open count you also need to close the library for every time
// that it is opened independently by the application as well.
#define  twMpEventGuestStartGame     3

// Host can initiate the game. All slaves should have their listeners
// and only the ACL links to devices still exist.  Comes with a list
// of addresses.
// IMPORTANT: the btlib is left open by the multiplayer library
// so that the ACL links can be left in place.  This means you do
// not have to call BtLibOpen to use the BtLib APIs.  A call to TwMpClose
// will close the BtLib for the multiplayer library.  Since the BtLib
```

```
// keeps an open count you also need to close the library for every time
// that it is opened independently by the application as well.
#define  twMpEventHostStartGame      4

// ARM aligned name structure
typedef struct _TwMpPlayerNameType
{
   UInt8* name;
   UInt32 nameLength;
} TwMpPlayerNameType;


#define twMpAskUserParamTypeVersion (1)
// Param structure passed into PilotMain sublaunch to replace
// invitation dialog.
typedef struct _TwMpAskUserParamType
{
   UInt32 version;
   TwMpPlayerNameType hostName;  //Name of game host.
} TwMpAskUserParamType;


#define twMpLaunchParamsTypeVersion (1)
// Structure stored in twMpFtrNumMultiplayerLaunch feature number
typedef struct _TwMpLaunchParamsType
{
   UInt32 version;
   // If fromCard is true then volRefNum and path are valid, otherwise not.
   Boolean fromCard;
   UInt8 reserved[3];  // padding
   UInt16 volRefNum;   // volume app was stored on
   UInt8 reserved1[2]; // padding
   const Char* path;   // path of the prc
} TwMpLaunchParamsType;

// Info on a specific player
typedef struct _TwMpPlayerInfoType
{
   UInt32 size;              // Caller MUST set this to sizeof(TwMpPlayerInfoType)
         TwMpPlayerStateEnum state;  // State of the player
         UInt32 reason;            // Reason if state is twMpPlayerFailed 16 bit error code, 32 for alignment
         TwMpPlayerNameType name;    // Name  of the player,NEEDS TO BE INITIALIZED PRIOR
                                     // TO CALLING TwMpGetPlayerInfo().
} TwMpPlayerInfoType;
```

```
typedef struct _TwMpEventType{

   UInt32 version;

   TwMpEventEnum event;

   union
   {
      // Event: twMpEventPlayerState
      struct
      {
         TwMpPlayerStateEnum newState; //updated player state

         //16 bit error code, 32 bits for structure alignment
         UInt32 reason; //reason for transition to a state (for twMpPlayerFailed)
         BtLibDeviceAddressType bdAddr;
      } state;

       // Event: twMpEventHostStartGame
      struct
      {
         // List of accepted players
         BtLibDeviceAddressType hostAddr;
         BtLibDeviceAddressType guestAddrs[kTwMpMaxOtherPlayers];
         UInt32 numGuests;
      } acceptedPlayers;

      // Event: twMpEventGuestStartGame
      BtLibDeviceAddressType hostAddr;

   }eventData;

} TwMpEventType;
```

PalmOS Launch Codes

sysAppLaunchCmdNormalLaunch

Purpose This is the standard PalmOS launch code.

Comments This launch code is mentioned because whenever an application that supports TwMp is launched it needs to check for the presence of the twMpFtrNumMultiplayerLaunch feature to determine if it was launched by TwMp or not. The feature stores a pointer to a TwMpLaunchParamsType structure (68K Big Endian byte order)

Header `SystemMgr.h`

twMpAppLaunchCmdAskUser

Purpose A game will be launched with this launch code to allow a game to replace the default invitation dialog with its own.

Parameters `cmdPBP` This argument to PilotMain points to a TwMpAskUserParamType structure containing the hosting device name.

Comments An app should only handle this launch code if it wants to replace the invitation UI. This is a sub-launch of PilotMain, so it should be treated like a subroutine rather than an asynchronous event. Put up UI to allow the user to accept or reject a game invitation. If the user rejects the invitation then PilotMain should return twMpResultDecline. If the user accepts the invitation then PilotMain should return twMpResultAccept. If twMpResultAccept is returned then the game is subsequently launched by TwMp with the sysAppLaunchCmdNormalLaunch launch code.

Header `TwMp.h`

API Calls

TwMpOpen

Purpose Open the multiplayer service

Prototype
```
Err TwMpOpen(void)
```

Result twMpErrNoError.

Comments Every call to TwMpOpen must have a corresponding call to TwMpClose or there is a risk of resource leak and the bluetooth library being left open causing bluetooth to not work properly for the rest of the system on application exit.

Header TwMp.h

Sample `TBD`

TwMpClose

Purpose Close the Multiplayer service

Prototype
```
Err TwMpClose(void)
```

Result twMpErrNoError

Comments Every call to TwMpOpen must have a corresponding call to TwMpClose or there is a risk of resource leak and the bluetooth library being left open causing bluetooth to not work properly for the rest of the system on application exit.

Header `TwMp.h`

Sample `TBD`

TwMpHostGame

Purpose Prepare to host a multiplayer game.

Prototype
```
Err TwMpHostGame(UInt32 creator, TwMpProcPtr callbackP, UInt32 refcon)
```

Parameters `creator` `[in]` The creator ID of the application that is hosting a multiplayer game.

`callbackP [in]` The callback function pointer for status updates about the multiplayer game formation process.

`refcon [in]` Reference context. User defined data to be passed as an argument to the callback function.

Result twMpErrNoError – No Error

twMpErrNotOpen - Multiplayer library not open.

twMpErrOutOfMemory – not enough memory to complete the call.

btLibErrBusy – the Bluetooth Library is in use by serial VDRV

btLibErrRadioInitFailed – Bluetooth initialization failure.

btLibErrFailed – BtLibPiconetCreate called with piconet already created.

Comments This function calls BtLibOpen and BtLibPiconetCreate. If you are using the bluetooth library then you must make sure that the piconet is destroyed and there are no existing ACL links to other devices. It is important to make sure that the callback function is declared using the SYSTEM_CALLBACK specifier to make sure your global variables are available in the callback function.

Header `TwMp.h`

Sample `TBD`

TwMpHostInvitePlayer

Purpose Invite a bluetooth device to play a game

Prototype
```
Err TwMpHostInvitePlayer(BtLibDeviceAddressType* bdAddrP)
```

Parameters `bdAddrP [in]` Bluetooth device address

Result twMpErrPending – Results will be returned via callback events.

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – bdAddrP is NULL

twMpErrNotHost – Only Hosts can call this API.

twMpErrTooMany – Already have the maximum number of connections in place or pending.

twMpErrOutOfMemory – not enough memory to complete the call.

twMpErrInProgress – A link is already in progress to this device..

twMpErrAlreadyConnected – This player has already accepted an invitation

Comments This call starts the process of adding a player to the game.

Events `twMpEventPlayerState – callback is called with this event whenever the state of a device changes.`

Header `TwMp.h`

Sample `TBD`

TwMpGetPlayerInfo

Purpose Get Information about a player.

Prototype
```
Err TwMpGetPlayerInfo(BtLibDeviceAddressTypePtr bdAddrP,
                      TwMpPlayerInfoType* infoP)
```

Parameters

`bdAddrP [in]`	Bluetooth device address.
`infoP [out]`	Pointer to Info structure. You must fill in the name structure with space for the name to be filled in.

Result twMpErrNoError – Info retreived successfully

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – Parameter error.

twMpErrNotFound – No information on that device.

Comments In the infoP structure you can set the friendly name pointer and length to zero and all the other data except the name will be returned without error.

Header `TwMp.h`

Sample `TBD`

TwMpGuestAcceptInvitation

Purpose Called by an application when it is launched by the Multiplayer service after a game invitation is accepted by a player to signal that the invitation has been accepted.

Prototype
```
Err TwMpGuestAcceptInvitation(TwMpProcPtr callbackP, UInt32 refCon)
```

Parameters

`callbackP [in]`	The callback function pointer for status updates about the multiplayer game formation process. .
`refCon [in]`	The time when the mute is canceled. Use zero to mute indefinitely.

Result twMpErrNoError – Invitation accepted. Future updates on all devices in the game will be updated through callback events.

twMpErrNotOpen - Multiplayer library not open.

twMpErrHost – Host cannot call this API

twMpErrNoConnection – No connection to host device.

Comments An application can tell whether it was launched by the multiplayer app if the twMpFtrNumMultiplayerLaunch exists on a sysAppLaunchCmdNormalLaunch. It is important to make sure that the callback function is declared using the SYSTEM_CALLBACK specifier to make sure your global variables are available in the callback function.

Events twMpEventPlayerState – callback is called with this event whenever the state of a device changes.

Header `TwMp.h`

Sample `TBD`

TwMpHostDisinvitePlayer

Purpose Host disinvites player from the game.

Prototype
```
Err TwMpHostDisinvitePlayer(BtLibDeviceAddressType* bdAddrP)
```

Parameters

`bdAddrP [in]`	Bluetooth device address.

Result twMpErrPending – Disinvite started, device updates will be returned through a callback.

twMpErrNotOpen - Multiplayer library not open.

twMpErrNotHost– Only Host can call this API.

twMpErrParamError – bdAddrP is NULL.

twMpErrInProgress – ACL link in progress cannot be canceled.

twMpErrDisconnected – Device already disconnected.

twMpErrHost – Attempted to disinvite yourself, not allowed.

twMpErrNotFound – Device not in the list of invited players.

Comments

Header `TwMp.h`

Sample `TBD`

TwMpHostStartGame

Purpose Start the game.

Prototype
```
Err TwMpHostStartGame(void)
```

Result twMpErrPending – Starting the game, updates sent through callback events

twMpErrNotHost – Only a host can call this API.

twMpErrNoAccepted – No accepted players, cannot start game.

Comments Guests will receive their final event twMpEventGuestStartGame to create listener sockets and advertise them. Then a the host will get a final callback with event twMpEventHostStartGame that has a list of the devices in the game.. These callbacks are synchronous so that there is no race condition. All the guests get their event twMpEventGuestStartGame before the host gets the twMpEventHostStartGame event

Events twMpEventPlayerState – callback is called with this event whenever the state of a device changes.

twMpEventGuestStartGame – Guests get this event when a game is starting. Upon receiving this event a guest should create a listener socket and advertise it for the game network. Guest should also register a management callback function to accept the handoff of the bluetooth network responsibility.

twMpEventHostStartGame – This event means that all hosts have received and processed the twMpEventGuestStartGame and the game can be started. The host should register a management callback function to accept the handoff of the bluetooth network responsibility. The host is responsible for initiating any L2Cap or RFComm connections at this point.

Header `TwMp.h`

Sample `TBD`

TwMpCancelGame

Purpose On a host, This call disconnects all devices in the piconet and cancels the game. On a guest this call removes the guest from the game.

Prototype
```
Err TwMpCancelGame(void)
```

Result twMpErrNoError – Game is canceled. Callback is unregistered

twMpErrPending – Game is in the process of canceling. Callbacks will cease after reception of twMpEventGameCanceled event.

twMpErrNotOpen - Multiplayer library not open.

Comments This call implicitly unregisters callbacks either immediately on twMpErrNoError or after reception of twMpEventGameCanceled on twMpErrPending.

Header `TwMp.h`

Sample `TBD`

TwMpGetNumPlayers

Purpose Get the number of players who have been invited into the game and also the host.

Prototype `Err TwMpGetNumPlayers(UInt8* numPlayers)`

Parameters

`numPlayers` `[out]`	Number of players invited or hosting the game.

Result twMpErrNoError

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – numPlayers is NULL

Comments

Header `TwMp.h`

Sample `TBD`

TwMpGetAllPlayers

Purpose Get the list of players who have been invited into the game and also the host.

Prototype
```
Err TwMpGetAllPlayers(BtLibDeviceAddressType bdAddr[], UInt8 arraySize,
                      UInt8* numPlayersReturnedP)
```

Parameters

`bdAddr` `[out]`	List of players who have been invited and the host.
`arraySize` `[in]`	Number of elements allocated in bdAddr array
`numPlayersReturnedP` `[out]`	Number of elements returned in bdAddr array

Result twMpErrNoError

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – illegal parameter passed in

twMpErrNotEnough – arraySize is too small to accept all data.

Comments

Header `TwMp.h`

Sample `TBD`

TwMpGetNumAcceptedPlayers

Purpose Get the number of players who have accepted an invitation into the game.

Prototype `Err TwMpGetNumAcceptedPlayers(UInt8* numPlayers)`

Parameters

`numPlayers` `[out]`	Number of players invited or hosting the game.

Result twMpErrNoError

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – numPlayers is NULL

Comments

Header `TwMp.h`

Sample `TBD`

TwMpGetAcceptedPlayers

Purpose Get the list of players who have accepted an invitation into the game.

Prototype

```
Err TwMpGetAcceptedPlayers(BtLibDeviceAddressType bdAddr[], UInt8
                arraySize, UInt8* numPlayersReturnedP)
```

Parameters

`bdAddr [out]`	List of players who have accepted a game invitation.
`arraySize [in]`	Number of elements allocated in bdAddr array
`numPlayersReturnedP [out]`	Number of elements returned in bdAddr array

Result twMpErrNoError

twMpErrNotOpen - Multiplayer library not open.

twMpErrParamError – illegal parameter passed in

twMpErrNotEnough – arraySize is too small to accept all data.

Comments

Header `TwMp.h`

Sample `TBD`

Callback Events

It is important to make sure that the callback function is declared using the SYSTEM_CALLBACK specifier to make sure your global variables are available in the callback function.

twMpEventPlayerState

Purpose A callback with this event happens every time there is a change in a players state.

Event Specific Callback Info

`newState`	New state of the device
`Reason`	Reason for transition to a state (for twMpPlayerFailed)
`Player`	Address of device that changed states.

Comments

Header `TwMp.h`

twMpEventHostStartGame

Purpose A callback with this event happens at the completion of the game starting process so that the host can start up the game. All the ACL links remain in place but the sockets have been torn down. This event is triggered by a call to TwMpHostStartGame(). No more callbacks will occur after this event is processed. Host should register the BtLib management callback upon getting this callback event.

Event Specific Callback Info

`hostAddr` Address of the host.

`guestAddrs` List of multiplayer game guests.

`numGuests` Number of addresses in guestAddrs list.

Comments IMPORTANT: the btlib is left open by the multiplayer library so that the ACL links can be left in place. This means you do not have to call BtLibOpen to use the BtLib APIs. A call to TwMpClose will close the BtLib for the multiplayer library. Since the BtLib keeps an open count you also need to close the library for every time that it is opened independently by the application as well.

Header `TwMp.h`

twMpEventGameCanceled

Purpose A game has been canceled. No more callbacks will occur. Triggered by a call to TwMpCancel() that returns twMpErrPending.

Comments IMPORTANT: the btlib is left open by the multiplayer library so that the ACL links can be left in place. This means you do not have to call BtLibOpen to use the BtLib APIs. A call to TwMpClose will close the BtLib for the multiplayer library. Since the BtLib keeps an open count you also need to close the library for every time that it is opened independently by the application as well.

Header `TwMp.h`

twMpEventGuestStartGame

Purpose This event is sent to each guest device when the Host has called TwMpHostStartGame(). The guest should register its management callback, create its listener socket, and advertise it upon receiving this event. No more callbacks will occur.

Event Specific Callback Info

`hostAddr` Address of the host.

Comments It is recommended that all connections are initiated by the host upon receiving the twMpEventHostStartGame event not this event. Initiating connections with this event in games with larger amounts of players could lead to a race condition where the game and TwMp use more bluetooth stack resources than are available.

IMPORTANT: the btlib is left open by the multiplayer library so that the ACL links can be left in place. This means you do not have to call BtLibOpen to use the BtLib APIs. A call to TwMpClose will close the BtLib for the multiplayer library. Since the BtLib keeps an open count you also need to close the library for every time that it is opened independently by the application as well.

Header `TwMp.h`

Player States

twMpPlayerHosting

Purpose This player is hosting the game.

Comments

Header `TwMp.h`

twMpPlayerWaiting

Purpose Waiting for bluetooth connection to get in place

Comments

Header `TwMp.h`

twMpPlayerAclConnected

Purpose Connected to device looking for multiplayer service

Comments

Header `TwMp.h`

twMpPlayerConnected

Purpose Connected to service. Player being asked to join the game.

Comments

Header `TwMp.h`

twMpPlayerAccepted

Purpose Player has accepted the game invitation.

Comments

Header `TwMp.h`

twMpPlayerDeclined

Purpose Player has declined the game invitation.

Comments

Header `TwMp.h`

twMpPlayerFailed

Purpose Connection to bluetooth device failed

Comments Refer to the reason returned with the twMpEventPlayerState callback or in TwMpPlayerInfoType structure as a result of calling TwMpGetPlayerInfo for a more specific error code.

Header `TwMp.h`

API Usage

This API is used differently depending on whether you are Hosting the game or a Guest participant.

Host Only APIs

TwMpHostGame
TwMpHostInvitePlayer
TwMpHostDisinvitePlayer
TwMpHostStartGame

Guest Only APIs

TwMpGuestAcceptInvitation

General APIs

TwMpOpen
TwMpClose
TwMpCancelGame
TwMpGetPlayerInfo
TwMpGetNumPlayers
TwMpGetAllPlayers
TwMpGetNumAcceptedPlayers
TwMpGetAcceptedPlayers

Example game formation process

Some of the multiplayer APIs are used differently if you are a game host or a game guest. A host would start a game by with the following basic steps.

1. Host starts the game

2. Discover the players to invite. One way to do this is to use the BtLibDiscoverMultipleDevices call. A custom discovery process using the BtLib could also be used.
3. Call TwMpOpen().
4. Call TwMpHost().
5. Call TwMpHostInvitePlayer(). to invite each guest.
6. Monitor callback to determine who has accepted, declined, and failed.
7. Call TwMpHostStartGame() when all the appropriate guests have accepted their invitation.
8. Upon receiving twMpEventHostStartGame register BtLib management callback and connect to the listening guests via L2Cap or RFCOMM (ACL links are still in place).
9. When the multiplayer game ends call TwMpClose().

A guest would follow these basic steps.

1. Game is sublaunched with twMpAppLaunchCmdAskUser launch code to allow replacement of invitation dialog.
2. User accepts invitation to game. (If twMpAppLaunchCmdAskUser launch code is handled by the game to replace the invitation dialog then return twMpResultAccept from PilotMain to accept the invitation).
3. Game is launched and twMpFtrNumMultiplayerLaunch feature exists to indicate that game was launched by user accepting an invitation.
4. Call TwMpOpen().
5. Call TwMpGuestAcceptInvitation().
6. Monitor callback to determine who has accepted, declined, and failed.
7. Upon receiving twMpEventGuestStartGame register the BtLib management callback and also create and advertise a listener socket.
8. When the multiplayer game ends call TwMpClose()