



SCE-RT SDK Medius Game Communication Library (MGCL) Release 2.10

Overview

March 2005

© 2005 Sony Computer Entertainment Inc.

Publication date: March 2005

Sony Computer Entertainment Inc.
2-6-21, Minami-Aoyama, Minato-ku
Tokyo 107-0062, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

The *SCE-RT SDK Medius Game Communication Library (MGCL) Release 2.10 – Overview* is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *SCE-RT SDK Medius Game Communication Library (MGCL) Release 2.10 – Overview* is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® license agreements.

Unauthorized reproduction, distribution, lending, rental or disclosure of this book to any third party, in whole or in part, is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® license agreements.

Ownership of the physical property of the book is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion of the book, in whole or in part, its presentation, or its contents is prohibited.

The information in the *SCE-RT SDK Medius Game Communication Library (MGCL) Release 2.10 – Overview* is subject to change without notice. The content of this book is Confidential Information of Sony Computer Entertainment.


“” and “PlayStation” are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

Table of Contents

About This Manual	v
Changes Since Last Release	v
Typographic Conventions	v
Developer Support	v
Sony Computer Entertainment America (SCEA)	v
Sony Computer Entertainment Europe (SCEE)	vi
Introduction	1
Linking the Medius Game Communication Library	2
Initial Authentication and Registration	3
Continuous MGCL Game Server Operation	5
Server-Initiated World Creation	6
Disconnection and Shutdown	7
Integrated Server	8
Peer-to-Peer Host	9
Peer-to-Peer Host Migration	11

This page intentionally left blank.

About This Manual

This manual provides a description of the various functionalities of the Medius Game Communication Library (MGCL). This API is an add-on to the DME API and Medius API provided by Sony Computer Entertainment America (SCEA).

Please forward any questions about this document to Greg Becksted (greg_becksted@playstation.sony.com).

Changes Since Last Release

- The **MGCLCreateGameOnMeRequest()** function now takes a **MGCLCreateGameOnMeRequestInParams** structure instead of **MediusServerCreateGameOnMeRequest**.
- **MGCLServerDisconnectPlayerCallback** has been added to **MGCLInitializeInParams** to help facilitate the Medius API function **MediusVoteToBanPlayer()**.

Typographic Conventions

Certain typographic conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
<code>courier</code>	Indicates literal program code.
<i>italic</i>	Indicates names of arguments and structure members (in structure/function definitions only).
medium bold	Indicates data types and structure/function names (in structure/function definitions only).
blue	Indicates a hyperlink.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: scert-support@scea.com scea_support@ps2-pro.com Web: https://www.ps2-pro.com/ https://psp.scedev.net Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees only in the PAL television territories (including Europe and Australasia). You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
Attn: Production Coordinator Sony Computer Entertainment Europe 13 Great Marlborough Street London W1F 7HP, U.K. Tel: +44 (0) 20 7859-5000	E-mail: scee_support@ps2-pro.com Web: https://www.ps2-pro.com/ https://psp.scedev.net Developer Support Hotline: +44 (0) 20 7911-7711 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT/BST)

Introduction

The Medius Game Communication Library (MGCL) provides the application developer the ability to post information about Internet-based games on the Medius platform. This information is used by the Medius Client to join existing games that are either in progress or in staging.

The Medius platform includes the following elements:

- Medius Universe Manager (MUM)
- Medius Authentication Server (MAS)
- Medius Lobby Server (MLS)
- Medius Proxy Server (MPS)

The MGCL complements the DME API and the Medius API by providing additional authentication and reporting features to the application. The application can then function as a server, while integrated into the overall DME architecture and the Medius platform.

An application with server functionality is considered to be an *MGCL game server*. The application must integrate with the MGCL for Internet-based games in the following situations:

- If it is a customized, stand-alone, Internet game server
- If it is an integrated server game, hosted on a user's computer
- If it is peer-to-peer game, hosted on a user's computer

Linking the Medius Game Communication Library

An application using the MGCL must already have the DME API library file linked to the final executable. If the application is being developed as an Integrated Server or as a peer-to-peer application with Medius lobby functionality, the Medius API library file must also be linked.

Function prototypes are available within `mgcl.h`. This file must be included in the main application source.

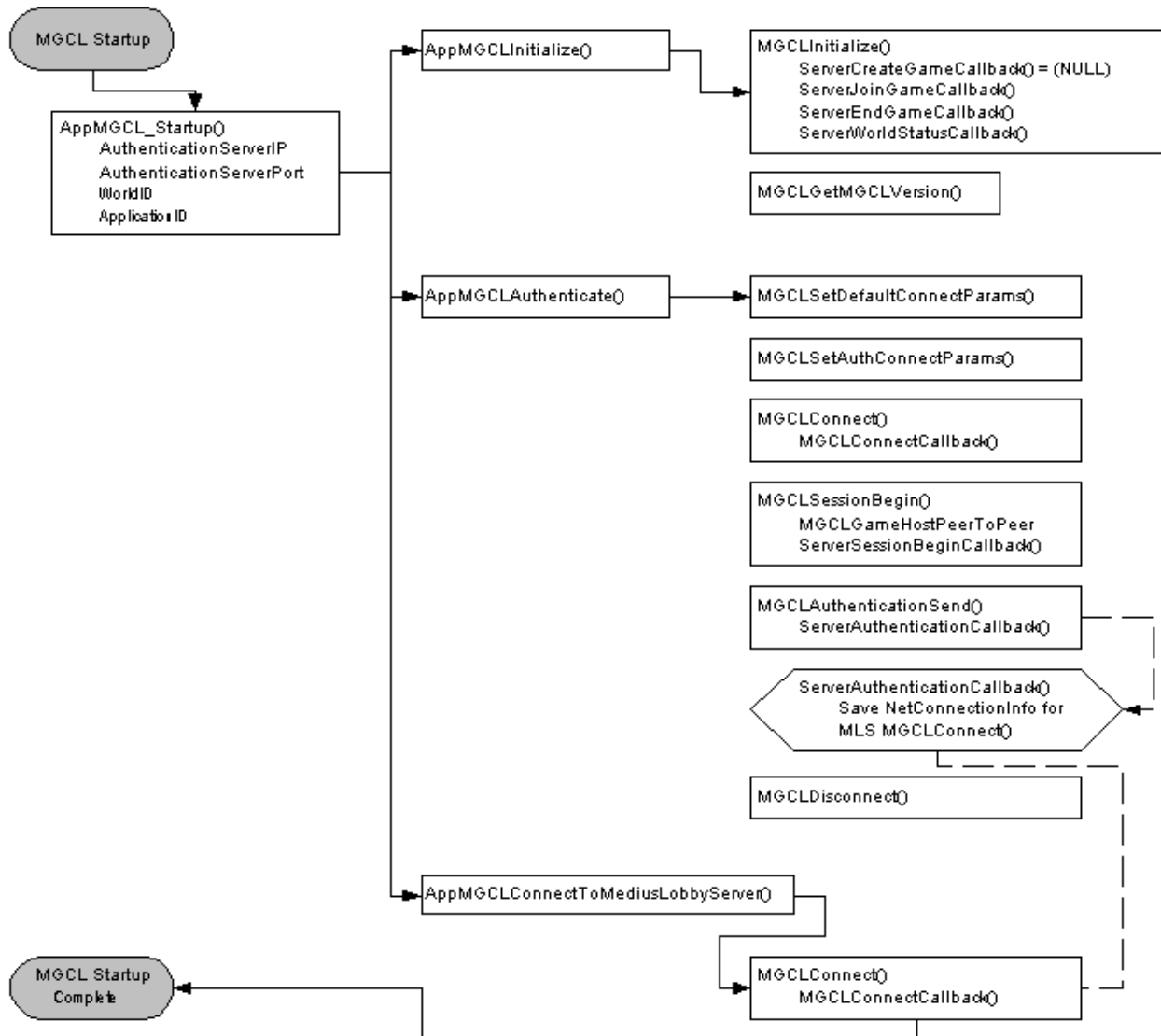
Initial Authentication and Registration

Figure 1 is a flow chart that shows the sequence of operations that must be performed to initiate an MGCL connection. As shown in the chart, the console must register and authenticate with the Medius platform before an application can function as a game server (stand-alone, integrated, or peer-to-peer), available for gameplay.

Figure 1: MGCL Startup Process

MGCL Flow Chart

MGCL Startup



Note: This flow chart shows the MGCL startup process from a high level. Functions that start with "App" are functions that wrap the MGCL API as used in the SCE-RT samples.

The sequence for initiating an MGCL connection is as follows:

1. Call **MGCLInitialize()** to start the MGCL engine. This normally assumes that the DME has already been initialized independently.
2. Initialize the **MGCLConnectInParams** information with the helper functions **MGCLSetDefaultConnectInParams()** and **MGCLSetAuthConnectParams()**.
3. Call **MGCLConnect()**, referencing the appropriate Medius Authentication Server (MAS) origination point. This IP/Port/World address information combination is provided directly by SCEA (the SCE-RT group).
4. Use **MGCLSessionBegin()** to begin a “session” with the Medius platform, which acts similarly to establish a cookie session with a an Internet browser.
5. Specify what “server type” (**MGCLGameHostClientServer**, **MGCLGameHostIntegratedServer**, or **MGCLGameHostPeerToPeer**) for this application’s MGCL connection while starting this “session”. All subsequent calls must also be followed by **MGCLUpdate()** to actually trigger the transmission and receipt of network traffic.
6. The callback registered to **MGCLSessionBegin()** will return a *SessionKey* as part of the **MediusServerSessionBeginResponse** in the **NetConnectionInfo** field, which should be stored for all future requests.
7. The game server must then call **MGCLAuthenticationSend()** with the appropriate data. This returns success or failure to the appropriate callback, along with a valid **NetConnectionInfo** type that contains Medius Lobby Server (MLS) IP/Port/World address information and an *AccessKey*. This **NetConnectionInfo** information tells the game server where to go next. Proceed with a call to **MGCLDisconnect()**, followed by another call to **MGCLConnect()** with the new key and address information.

Continuous MGCL Game Server Operation

Once the game server has completed the Initial Authentication and Registration steps, it then communicates with the Medius platform regarding incoming players and current status.

1. **MGCLUpdate()** must be called as often as possible (preferably on every frame) to ensure smooth network traffic to and from the client. This is the only way for data to actually be sent or received from the application.
2. **MGCLSendServerReport()** must be populated and sent approximately every 30 seconds or less. This provides the Medius platform with the current state of the game server, and allows for proper distribution of games and players across all game servers.
3. The Medius platform will occasionally contact the game server via the callbacks registered in **MGCLInitialize()**. These callbacks must determine whether a player wishes to create a game or join a game, or whether the Medius platform is attempting to end a game or query a game's status. These responses can be sent via **MGCLCreateGameResponse()**, **MGCLJoinGameResponse()**, **MGCLWorldStatusResponse()** and **MGCLEndGameResponse()**.

Server-Initiated World Creation

The following sequence allows a game server to create/initiate game instances (as opposed to players doing so).

1. Call **MGCLCreateGameOnMeRequest()** to identify the world and parameters of the game being created.
2. If the Medius platform responds successfully and favorably, call **MGCLSendServerReport()** with the appropriate game-specific parameters every 30 to 60 seconds. If this is not done in a timely fashion, the game instance will eventually time out.
3. Call **MGCLEndGameOnMeRequest()** to signal the graceful discontinuation of the game instance. This does not stop the game server operations. The Medius platform will continue waiting for the game server's MGCL reports in a normal fashion.

Disconnection and Shutdown

The following sequence must occur when the application is ready to shut down all game server operations.

1. Call **MGCLSessionEnd()** to gracefully terminate the session key with the Medius platform.
2. Call **MGCLDisconnect()** to disconnect from the network.
3. Call **MGCLClose()** to stop the MGCL game server engine. This does not stop the DME, so normal client operations will continue.

Integrated Server

The following sequence starts an integrated server. It will ensure that the game server is in the same location as the player.

1. Call **MediusSessionBegin()** to start a new 'player' session.
2. Call **MediusGetLocations()** to retrieve a list of all locations. Allow the player to select an entry from this list.
3. Get player input and call **MediusPickLocation()** to assign a *LocationID* to the player session.
4. Continue the player login process as before.
5. To begin activation of the Integrated Server, populate the **MediusServerSessionBeginRequest** structure (setting *LocationID* to the same value selected by player in Step 3, and *ServerType* set to **MGCLGameHostIntegratedServer**) and call **MGCLSessionBegin()**.
6. Continue the server authentication process as before.

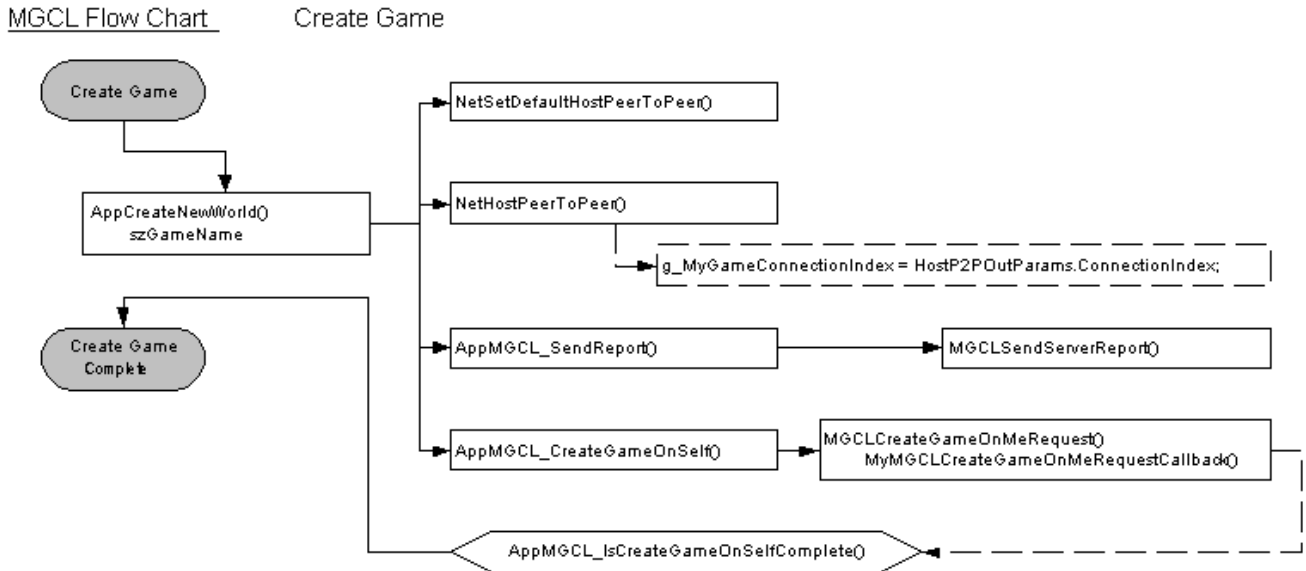
The following sequence switches to another Integrated Server when a player decides to change locations. It will ensure that the game server is in the same location as the player.

1. Call **MGCLSessionEnd()** to end the current integrated server session.
2. Call **MediusGetLocations()** to retrieve a list of all locations. Allow the player to select an entry from this list.
3. Call **MediusPickLocation()** to assign a new *LocationID* to the player session.
4. Call **MGCLSessionBegin()** using a new *LocationID* in **MediusServerSessionBeginRequest** structure.
5. Continue the server authentication process as before.

Peer-to-Peer Host

Figure 2 is a flow chart that shows the sequence of operations that must be performed to start a peer-to-peer host and create a game.

Figure 2: Create Game Process



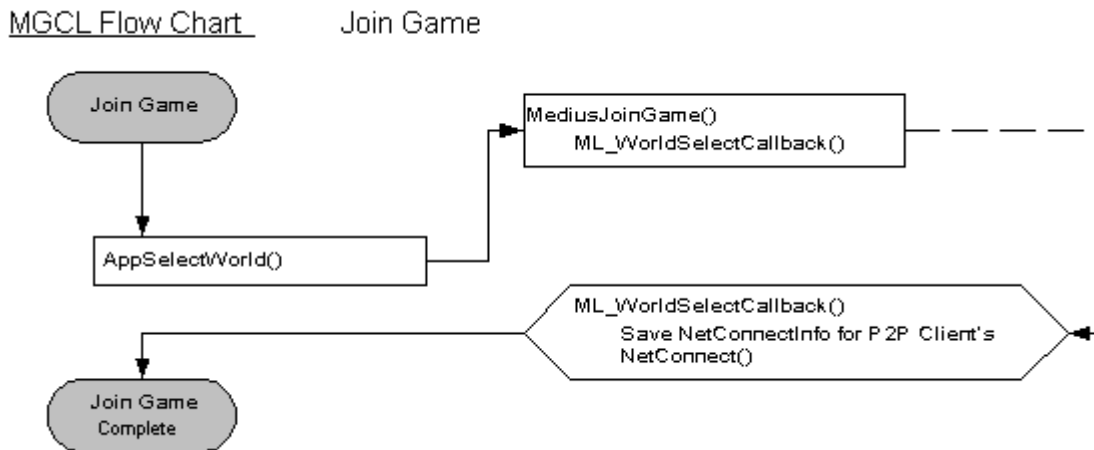
Note: This flow chart shows the MGCL Create Game process from a high level. Functions that start with “App” are functions that wrap the MGCL API as used in the SCE-RT samples.

The sequence for starting a peer-to-peer host and creating a game is as follows:

1. After a player has logged in with the Medius platform, that player has the option to either create or join a peer-to-peer game from the application’s lobby. To create a game, an MGCL connection on the player’s machine **must** have been previously established (see Figure 1). To initiate an MGCL connection, refer to “Initial Authentication and Registration”. Keep in mind that *ServerType* = **MGCLGameHostPeerToPeer** is passed in when **MGCLSessionBegin()** is called (via the **MediusServerSessionBeginRequest** structure).
2. Assuming an active MGCL connection, execute the DME function call **NetHostPeerToPeer()** to create a game (see Figure 2).
3. Initialize the **NetHostPeerToPeerInParams** information with the helper function **NetSetDefaultHostPeerToPeerParams()**. **NetHostPeerToPeer()** will, in effect, have placed the application in a state where it is now ready for clients to start making connections.
4. Send a manual MGCL report (via **MGCLSendServerReport()** with *TotalActivePlayers* set to 1) to inform the Medius platform that our DME host has been created.
5. Call **MGCLCreateGameOnMeRequest()** to allow other players to see and join the game from the application’s lobby (via the Medius platform). This MGCL function will inform the Medius platform that there is a host available and ready for clients to start connecting. If a player requests a game list while in the application’s lobby, the Medius platform will show the host’s game in the returned list and other players will be able to join the game.

6. After the host has completed creating the game, the following reports must be issued periodically: **MGCLSendServerReport()**, **MediusSendWorldReport()**, **MediusSendPlayerReport()**, and **MediusUpdateClientState()**. If the host fails to send these reports, the Medius platform may assume the host or some of the players in the current game have timed out. If the Medius platform ever believes that the game or players have timed out, the game can still continue. However, the Medius platform will not have up-to-date information about the game in progress and/or information about the players in the game (for example, current stats).
7. A player wishing to join a peer-to-peer game does not need to know anything about MGCL. Once the player has logged in to the Medius platform, all that is required is to request a game list from the application's lobby and, as shown in Figure 3, execute the Medius Client function call **MediusJoinGame()** with the *MediusWorldID* of the game they wish to join. At this time, the Medius platform will be sending joining connection information to the host's MGCL game server connection and back to the player wishing to join. If the host's MGCL game server callback **MGCLServerJoinGameCallback()** responds with success back to the Medius platform, it will forward the host's DME connection information back to the joining player via the **MediusJoinGame()** callback so that the player can then do a separate **NetConnect()** to the host of the game at the DME level.

Figure 3: Join Game Process



Note: This flow chart shows the MGCL Join Game process from a high level. Functions that start with “App” are functions that wrap the MGCL API as used in the SCE-RT samples.

8. The host is now ready to send a **MGCLSendServerReport()** as players connect and disconnect from the game (keeping the *TotalActivePlayers* current). This keeps the Medius platform informed of the current number of players in a game.

Peer-to-Peer Host Migration

Developers should consider if host migration is an important element of a title, as a host can leave a game at any time, either gracefully or ungracefully. There are two options: all players can be dropped from the game and returned to the lobby, or hosting responsibilities can be migrated to another player.

Using the DME/MGCL/Medius platform architecture makes it easy to detect when a host needs to migrate and to ensure that hosting responsibilities migrate to another player with minimal discrepancy of gameplay.

If host migration is needed for a title, the developer must also decide how and when players joining a peer-to-peer game already in progress will initialize and startup their own MGCL connections. One option is to have all players create their MGCL connection before they actually create or join a game.

The benefit of all players having an established MGCL connection is obvious. When the DME detects that the host has left the game, the process to initiate host migration can occur fairly quickly. Otherwise, in situations where only the original host has a MGCL connection, the current game (in all likelihood) must halt as a new host must go through the whole process of starting up a MGCL connection (as described in “Initial Authentication and Registration” on page 3). This can be a time-consuming process.

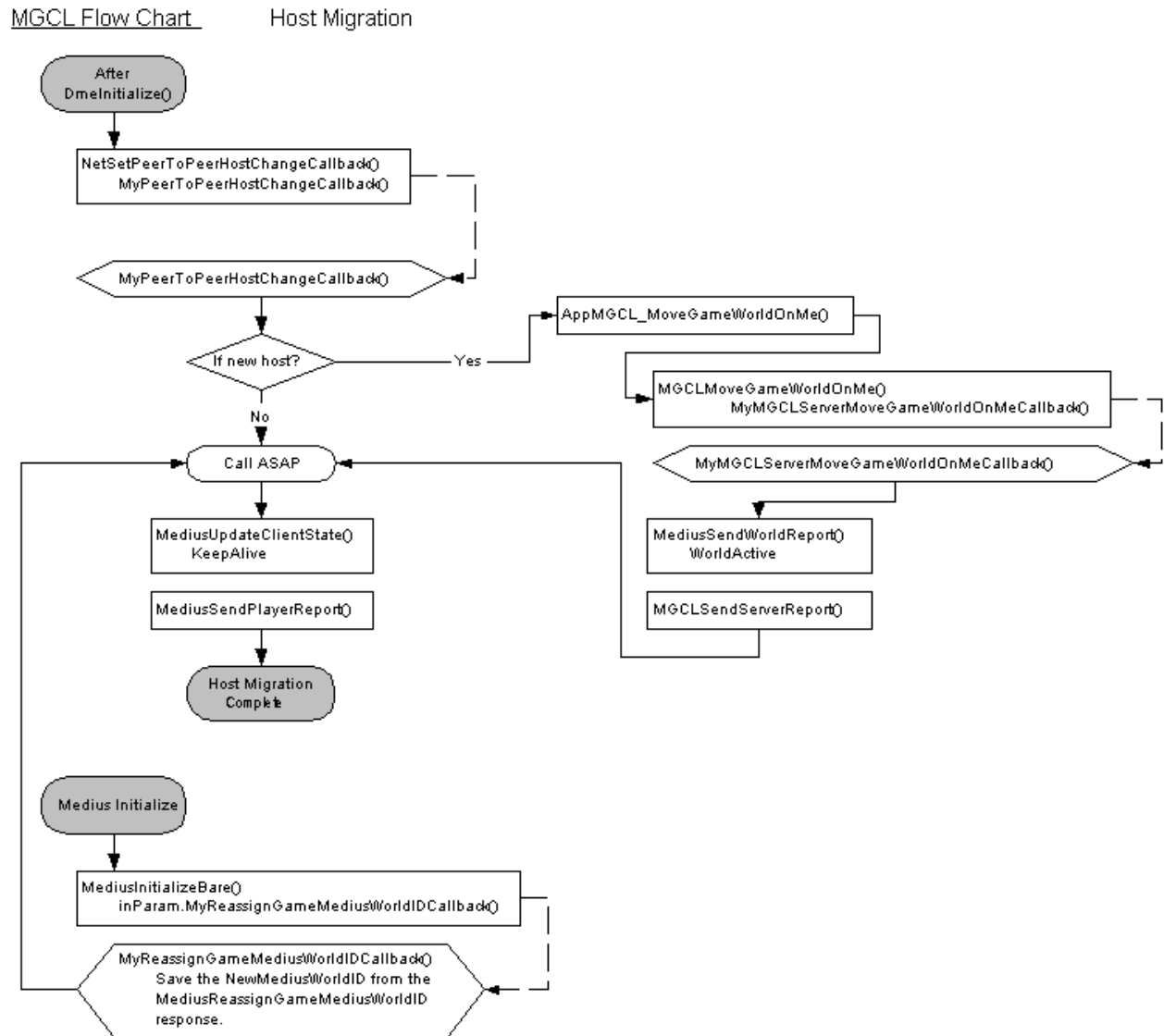
Figure 4 is a flow chart that shows the sequence of operations that must be performed to migrate a host.

The sequence for migration of a host is as follows:

1. Calling **NetSetPeerToPeerHostChangeCallback()** registers the DME host migration callback function. Thus, if the current host leaves a game (either gracefully or ungracefully) the DME will let all players know which player has become the new host.
2. The callback function associated with **MediusInitializeBareInParams** called **MediusTypeReassignGameMediusWorldIDCallback()** is registered during **MediusInitializeBare()**. This callback function registers the *MediusWorldID* change callback. When a host migration takes place, the new host must call **MGCLMoveGameWorldOnMe()**. The Medius platform will then assign a new *MediusWorldID* that all players will need for their MLS and MGCL reports (i.e. **MGCLSendServerReports()**, **MediusSendWorldReports()**, **MediusSendPlayerReports()**, and **MediusUpdateClientState()**). Every player should reissue updated reports to the Medius platform upon receiving the new *MediusWorldID*.
3. The player who has been designated as the new host must have an active MGCL connection; otherwise, the new host will need to establish a new MGCL connection. Once a MGCL connection has been verified, the new host will then call **MGCLMoveGameWorldOnMe()** to request ownership of the game. If the Medius platform responds with a successful confirmation, a new *MediusWorldID* will be issued to all players and the new host will be responsible for maintaining the state of the current game.

If the Medius platform responds with an unsuccessful confirmation, the current game will be unaffected and will still be ongoing. However, the Medius platform’s knowledge of the game will eventually time out. It will no longer have information about the game’s status and/or of the players in that game. The Medius platform will no longer be accepting world, player, and/or MGCL reports from a game that has timed-out.

Figure 4: Host Migration Process



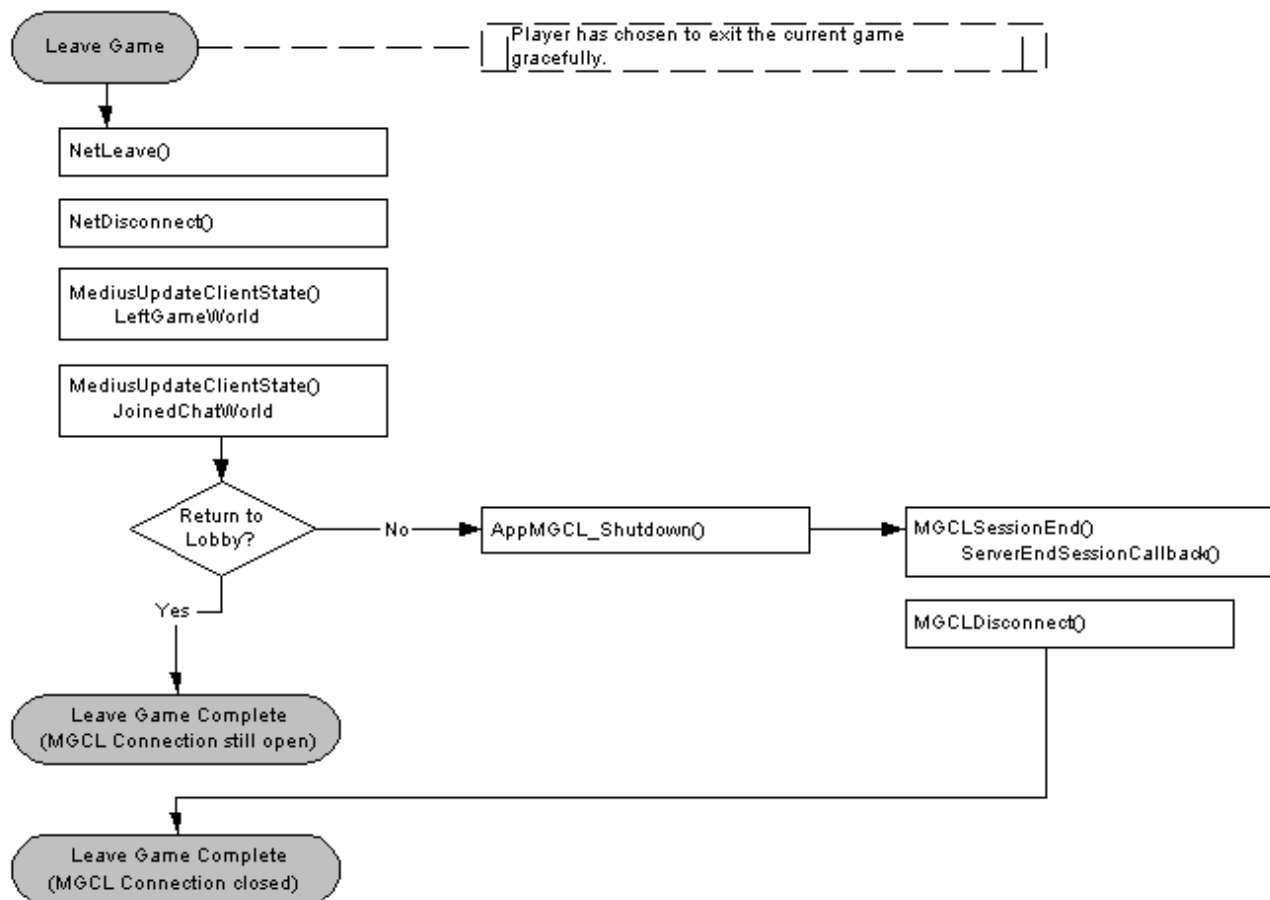
Note: This flow chart shows the MGCL Host Migration process from a high level. Functions that start with “App” are functions that wrap the MGCL API as used in the SCE-RT samples.

4. If either a host or player leaves a game to return to the application’s lobby, **MediusUpdateClientState()** must be called two times: the first time with *MediusUserAction* set to ‘LeftGameWorld’, and the second time with *MediusUserAction* set to ‘JoinedChatWorld’. If this does not occur, it may not be possible to retrieve game or player lists from the Medius platform. Figure 5 shows this process.

Figure 5: Leave Game Process

MGCL Flow Chart

Leave Game



Note: This flow chart shows the MGCL Leave Game process from a high level. Functions that start with “App” are functions that wrap the MGCL API as used in the SCE-RT samples.

This page intentionally left blank.