Game Services Architecture

Approved Version 1.0 - 29 Mar 2011

Open Mobile Alliance

OMA-AD-Game-Services-V1_0-20110329-A

Contents

Figures

1. Scope (Informative)

This document describes the base concepts and high-level architecture of the OMA Gaming Service. Technical details of the server framework are covered in the specifications for Gaming Platform 1.0 [GP10] and the technical details for the client and server are covered in Client/Server Interface [ClientServerInterface].

Starting from business drivers, the document describes the key features of the OMA gaming platform in enabler release 1.0 and enabler release 2.0 and sets them into perspective.

2. References

2.1 Normative References

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.3, Open Mobile Alliance™, OMA-IOP-Process-V1_3, URL:http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt

[GP10] OMA-GS-v1_0, http://www.openmobilealliance.org/release_program/enabler_releases.html

[GP10JDOC] OMA-GamingPlatform-JavaDocs-V1_0, http://www.openmobilealliance.org/release_program/enabler_releases.html

[GSRD] "OMA Game Services Requirements", Version 1.0, Open Mobile Alliance™, OMA-RD-Game-Services-V1_0, URL:http://www.openmobilealliance.org/

[IGCOMRD] "OMA Game Services/In Game Communication Requirements", Version 1.0, Open Mobile Alliance™, V1_0_0, URL:http://www.openmobilealliance.org/

[GSARCH] "OMA Game Services Architecture ", Version 1.0, Open Mobile Alliance™, V0_0_1, URL:http://www.openmobilealliance.org/

[DLOTA] Generic Content Download Over The Air Specification, Version 1.0, OMA-Download-OTA-v1_0, http://www.openmobilealliance.org/

[ClientServerInterface] "OMA Game Services Client/Server Interface" Version 1.0, Open Mobile Alliance™, OMA-TS-Game-Services-Client-Server-Interface-V1_0, URL:http://www.openmobilealliance.org/

2.2 Informative References

[JAVA] "The Java Language Specification", http://java.sun.com

[J2ME] JSR-68 "J2ME Platform Specification" JCP, http://www.jcp.org/en/jsr/all

[MIDP10] JSR-37 "Mobile Information Device Profile for the J2ME Platform" JCP, http://www.jcp.org/en/jsr/all

[MIDP20] JSR-118 "J2ME Mobile Information Device Profile" JCP, http://www.jcp.org/en/jsr/all

[MSNET] "Getting Started with Visual Studio .NET and the Microsoft .NET Compact Framework", http://msdn.microsoft.com

[OMAVOCAB] Dictionary for OMA Specifications V2.3, URL:http://www.openmobilealliance.org/

[WIPI] KWISFS.K-05-002 "Wireless Internet Platform for Interoperability" URL:http://wipi.or.kr/English/

[IMSArch] "Utilization of IMS capabilities Architecture", OMA-AD-IMS-V1_0 URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

This is an informative document, which is not intended to provide testable requirements to implementations.

3.2 Definitions

Activate	Activating an *ActorSession* results in resuming the *ApplicationInstance* at the previous state.
Actor	Represents the persistent data for the ActorSession.
Actor Number	An internal numbering of *Actors, which* connects the *Actor* to an *ApplicationInstance.* This number is unique per *ApplicationInstance.*
ActorSession	Representation of a User in an ApplicationInstance. One user can have many Actors.
Administration	The task of modifying the behaviour of the product after it has been released and is in operation. The actor is usually other than System Integrator or Developer e.g. Administrator or Customer using administration tools.
API	API, or Application Programming Interface, is the interface by which an application program accesses operating system, platforms or other system services. An API provides a level of abstraction between the requesting application and the provider of the service, ensuring portability of the code. API main task is the translation of parameter lists from the calling format to the service provider format, while supporting the interpretation of call-by-value and call-by-reference arguments in one or both directions.
Application pull	Service type where an application gets data by requesting it from a mobile device.
Application push	Service type where an application sends data to a mobile device.
ApplicationInstance State	Define a phases of operation of an *ApplicationInstance.* Every *ApplicationInstance* can have one of three different states: *SetupState*, *RunningState*, *EndState* of operations. Operations may have a different or no effect depending on the *ApplicationInstance State*
Browser client	A device that contains software that allows a user to view or "browse" text-based or multimedia information on the Internet.
Client/Server interface	The client/server Interface specifies the API between the client and the server.
Client/Server contract	The client/server contract specifies the communication between the client and the server.
Competition	An event where Users compete against each other or try to reach a set objective. Competitions may last for a defined period of time. Usually some sort of prizes is awarded for the competitors based on their Scores.
Configuration	The task of modifying the behaviour of the product as a separate task from programming. Configurations are done as part of the development cycle, integration and administration. The behaviour of the product is a very wide-ranging term covering: user interface, logic, storage of data, performance, session management, logging etc.
Customer	The organization providing the product for the End-Users. Customisation is done according to the Customer requirements. For products offering mobile services, the Customer is often an operator or a portal owner.

Deactivate	Deactivating an *ActorSession* results in putting the *ApplicationInstance* "on hold" for later use.
Declarative timers	Application timers (events) that can be declared (scheduled) to occur in a reoccurring fashion.
Delivery report	A message notifying the sender whether the recipient has received a previously sent message.
Domain Model	A Domain Model describes a framework and high-level structure of the gaming server lifecycle.
End-user, gamer, user, game player, player, customer, consumer.	The real human user using the service, i.e. playing a game.
Event	An action or occurrence detected by a program. Events can be user actions, such as clicking a mouse button or pressing a key, or system occurrences, such as running out of memory. Most modern applications, particularly those that run in Macintosh and Windows environments, are said to be event-driven, because they are designed to respond to events.
Executable Client	The executable client has local processing and storage capacity, e.g. J2ME clients. Executable clients facilitate applications that can be used also when the device is not connected to the network or out of coverage.
Game Device	A Device that supports the OMA Game Service Enabler.
Game Client	The part of the game that exists in the Game Device
Game Server	A game server is part of the game service, which can provide matchmaking, lobbying, registrations, highscores and gameplay functions.
Game Service	Game Service can be regarded as generic infrastructure providing common services to a range of game applications. The Game Service consists of Game Clients and Game Servers.
Game Service Provider	A service provider that provides game services.
Game Session	An instance of a game in progress (game is executing). The game client and the game server have to join into the instance of the game.
Device, **Handset,** **Mobile Device**	See [OMAVOCAB] for 'Device'.
Integration	The task of making a product ready for Production. This includes integrating the work done in software, user interface and content development. There is often also need for Configuration of the product according to customer and locale needs.
Interoperability	Interoperability refers to the capability for applications running on different computers to exchange information and operate cooperatively using this information. In other words, the ability to share data and services.
IMS Service Capability	The IMS Service Capability is a capability that provides a service to the upper layer, i.e. OMA Service Enablers. Service capabilities can be used as building blocks for service enablers, and services.
PascalCase	Naming Convention: The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. Sample: HandoverTurn
Portal	A service that offers a broad array of resources and services, such as e-mail, forums, search engines, entertainment, and on-line shopping malls.
Programmatic Events	Events that can be added (scheduled) programmatically via the event API.
Protocol	An agreed-upon format for transmitting data between two devices.
Request/reply paradigm	A common pattern of communication used by application programs, where a client sends a request message to a server, the server responds with a reply message, and the client blocks waiting for this response.

Score	A single comparable integer value representing how well a user has done in the game.
Score Table	A list of Scores of Users organized by pre-defined criteria.
Server Game Logic	Define the part of the game that reside on the server side and interact with the client games;
Session	A series of interactions encompassing the Actor's lifetime within the specific Application Instance.
Security, **Secure**	See [OMAVOCAB] for 'Security'
Transaction	An atomic unit of work that modifies data. A transaction encloses one or more program statements, all of which either complete or roll back.
User	See [OMAVOCAB] for 'User'. Specifically in the GS Domain Context, the user is one of the entities in the Domain Model. The user exists independently of any game. The user has a unique identification. The user can have multiple Actor's.

3.3 Abbreviations

OMA Open Mobile Alliance

ASID ActorSession ID

MMS Multimedia Message Service

SID Session ID

SMS Short Messaging Service

4. Introduction (Informative)

Due to enhanced device capabilities of regular mobile handsets and newly emerging devices, specifically for gaming, the market for mobile gaming is growing rapidly. On one hand 'phones that have the ability to play games' have become more powerful and today include color displays, large memory, as well as download and execution environments making game applications more fun and easier to access.

In the wireless data services segment, mobile gaming is emerging as a mass market, and often used as a time killer for everyone. 'Connected mobile gaming consoles' on the other hand are moving from traditional games industry into mobile networks. This segment is targeted to passionate gamers and sets new standards for mobile game play.

Mobile games are the fastest growing type of applications for mobile devices at this stage and can easily become the 'killer application' for third generation networks. Good game applications can create high stickiness and user involvement, as well as stimulate significant amount of network traffic and end user consumption of high value content.

Those business drivers transpire into a win-win situation for all involved parties in the value chain:

- For the user, the game experience increases, due to instant and pervasive interaction and much more exciting games than has been seen in the past.
- Network operators see gaming as a family of applications, which significantly drives up network traffic, offers opportunities to sell more high value content, and increases stickiness. Mobile data is regarded as the key driver for telecommunications services growth.
- Service providers provide applications via download through portals and create revenue this way.
- For game publishers and developers, a mass market is emerging because their products are accessible by a mass market. Mobile gaming is a complement, not a replacement, for console gaming. Wireless connectivity will add value through a mass-market channel of delivery.

When looking at mobile games, both from a game experience and architecture perspective, it is useful to distinguish between the following categories of games:

- ***Messaging based games***, which are games using SMS or MMS messaging protocols, and have their game logic purely on the server. Such games can only be played when users are connected to the mobile network.
- ***Mobile browser-based games,*** using for example a WAP or i-mode browser, also follow the thin client approach as described above.
- ***Downloadable offline games,*** where game logic is downloaded on the mobile device and is executing on the mobile device only.
- ***Downloadable online games,*** where game logic is downloaded and executed on the device, but here in conjunction with a game server that can be accessed over a wireless network.
- ***Externally delivered games,*** which are distributed physically, through various media (e.g. memory cards etc) and are executing on the mobile device in conjunction with a game server that is accessed over a wireless network.

Beyond those game types listed above, also offline games that are not downloaded to the handset and not connected to the network exist, and are out of scope for this specification.

The game types listed above differ in terms of server side and client side components that are involved, as well as the interaction over the wireless network. State-of-the-art (in 2004) is downloadable games and for some type of devices memory card based games. Connected games are strongly emerging, making mobile gaming more interactive than ever.

4.1 Target Audience

The target audience for this document includes but is not limited to the following:

- The Working Group(s) that will create specifications based on this subject matter
- Working Groups that need to understand the architecture of this subject matter
- Architecture Working Group
- Interoperability Working Group (e.g. for early analysis of interoperability requirements)
- Security Working Group
- Architects, developers and business experts aiming to get an overview of OMA's Gaming Services. This document explains the architecture on a high level and sets the context of the more detailed technical documents [GP10] and [ClientServerInterface].

4.2 Use Cases

The OMA Game Services Requirements [GSRD] describes in detail of all possible Game Service Enabler use cases for. In the following sections, the use cases are some of the simple examples to ease reader understand Game Service Enabler.

The section 4.2.1 describes the OMA game service features and relationship with other OMA Enablers.

The Section 4.2.2 covers some of the basic client/server functions (Login/Registration, random matchmaking, highscore) message flow.

The section 4.2.3 use cases illustrate a number of real-life examples of the Chinese and Korean game services may be used. Of course, the game service functionality is not limited to these examples.

4.2.1 Download Game with DRM and Charging

The following Figure 1illustrates an example of using Java game download use case. The use case is based on the Use Case 1 (Distribution and Rights Management, Obtain Game) [GSRD], where the client is capable of downloading and executing applications, and game service provider must be aware of handset capabilities. The download procedure is based on OMA Generic Content Download OTA [DLOTA].

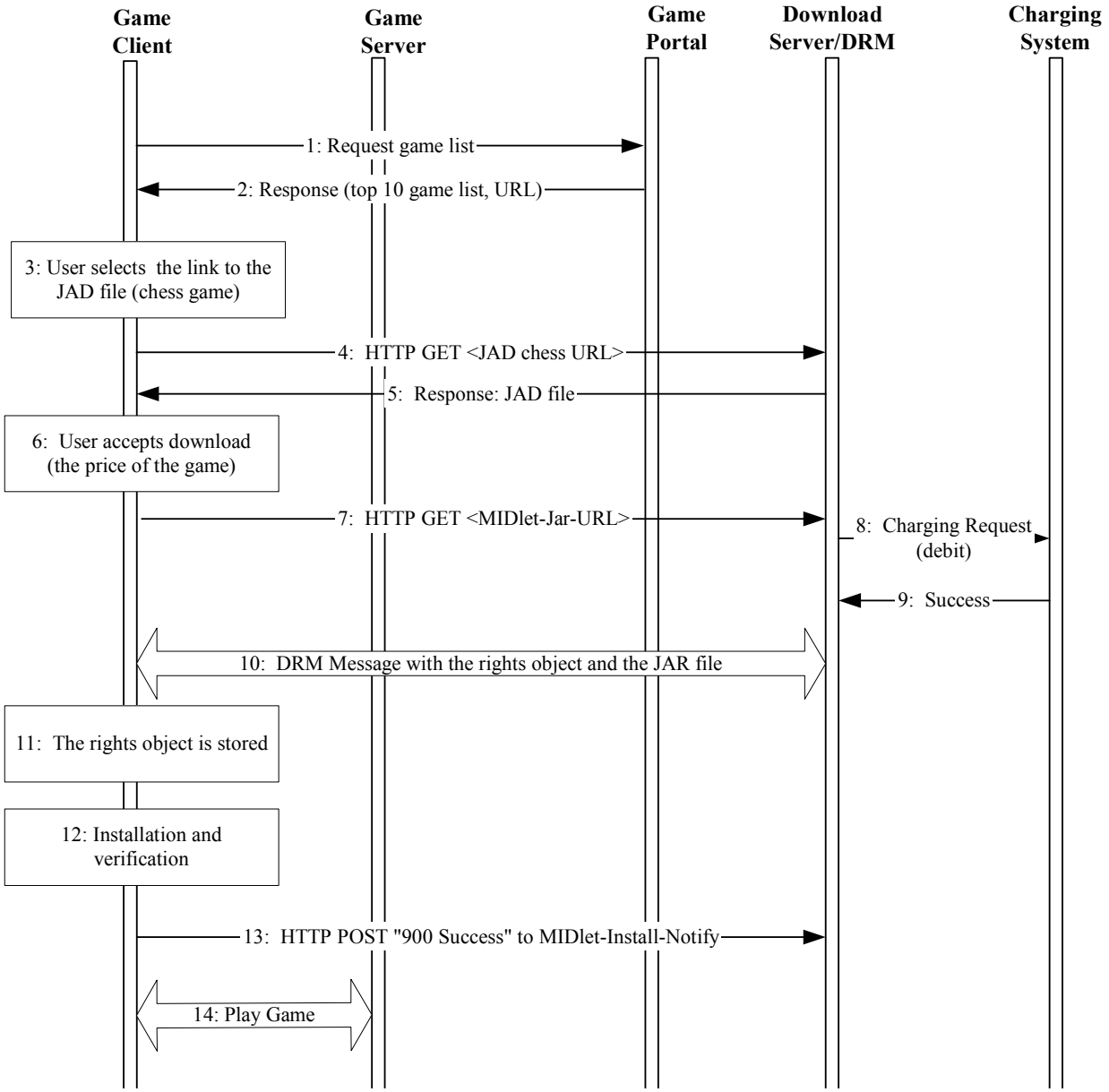

Figure 1: Example of Java game download and online charging.

Step 1-2: The user uses his client browser to enter the Game Service Provider's portal site and selects the link to a list of great games. The client displays only games, which are compatible with the client. Each listing shows the price will be charged.

Step 3: The user selects "chess" game.

Step 4: The client sends a HTTP GET with JAD-chess-URL to the game download/DRM.

Step 5: The game download/DRM responses with JAD file with DRM attributes.

Step 6: The user is asked to confirm to pay a certain amount. The user approves download of the MIDlet.

Step 7: The client uses a HTTP GET with URL from the MIDlet-Jar-URL attribute of the JAD file to request the "chess" game DRM message.

Step 8-9: Upon receiving the game requested, the game download/DRM server request credit from the charging system. If game installation would not be successful, then the money can be refunded to the charging system.

Step 10: The OMA DRM combined delivery method protected MIDlet has its JAR file wrapped into the DRM message, and accompanied with the rights object.

Step 11: After the DRM message is in the client, the rights object is stored.

Step 12: Then, the installation process continues with all necessary verifications are made.

Step 13: The client reports the installation status to the game download/DRM server.

Step 14: The game is stored in the user client, are ready play.

4.2.2 Client and Server Functions

4.2.2.1 Login/Registration

The following Figure 2illustrates an example of Login/Registration use case. The use case is based on the Use Case 4 User Participation, In-Progress Games (Joining an on-going game) [GSRD]. In this use case, it is assumed that the game client has all the information to precede the Login/Registration process, e.g. address of the game server.

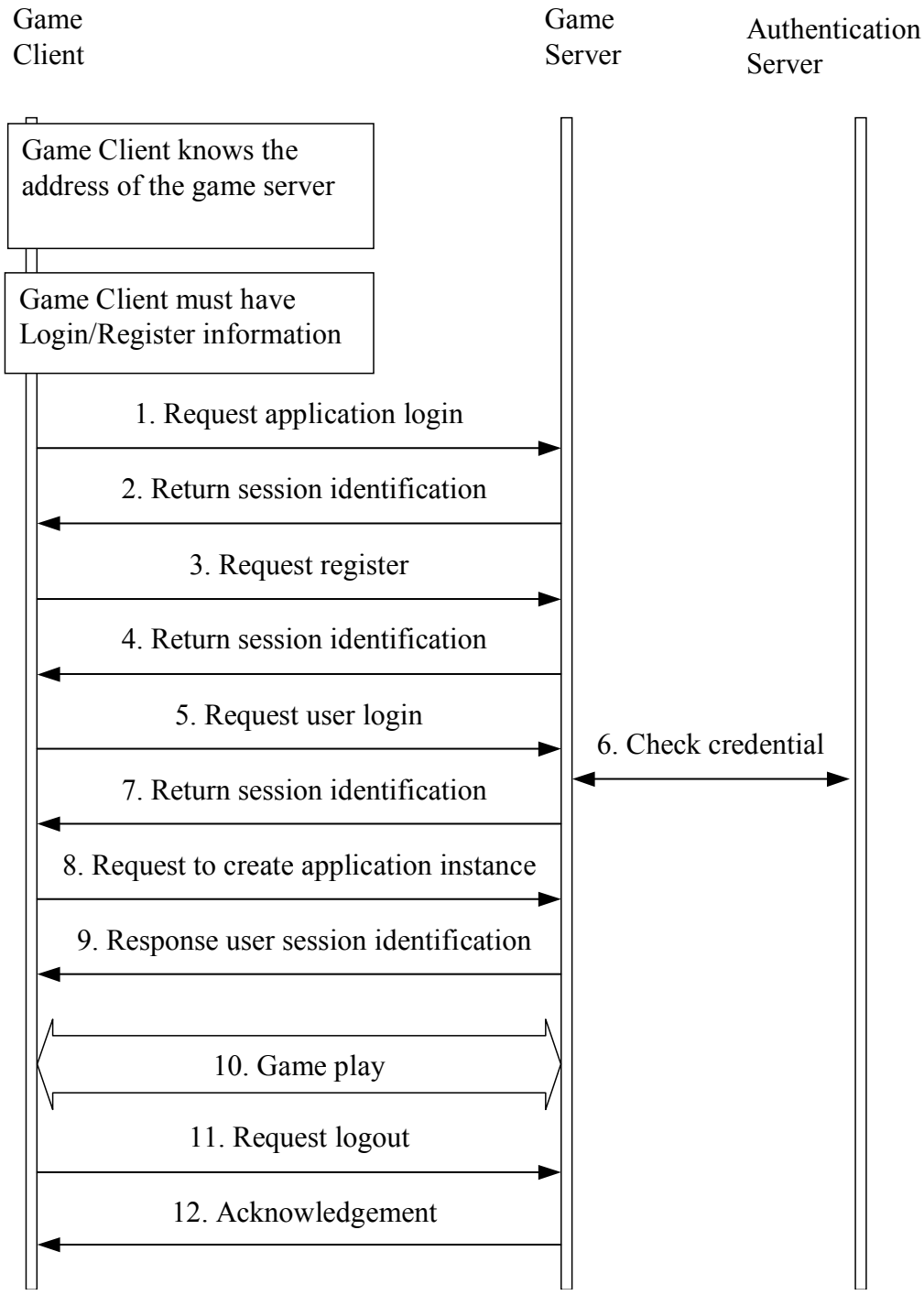

Figure 2: Example of Login/Registration

Step 1-2: The user uses his game client to login the game server. Most of all, the game client request application login using the game server address and the application login information that the game client already know.

Step 3-4: If the user is a new comer, performs registration process.

Step 5-7: During the login process, the game server requests the authentication server to check the credential of the user.

Step 8-9: The login/registration is success, the game client requests the game server to create application instance.

Step 10: The user enjoys his game.

Step 11-12: The user wants to stop the game. Therefore, the user uses the game client and requests the game server to logout the user.

4.2.2.2 Random Matchmaking

The following Figure 3illustrates an example of using random matchmaking use case. The use case is based on the Use Case 3 User Participation, Player Matching [GSRD].

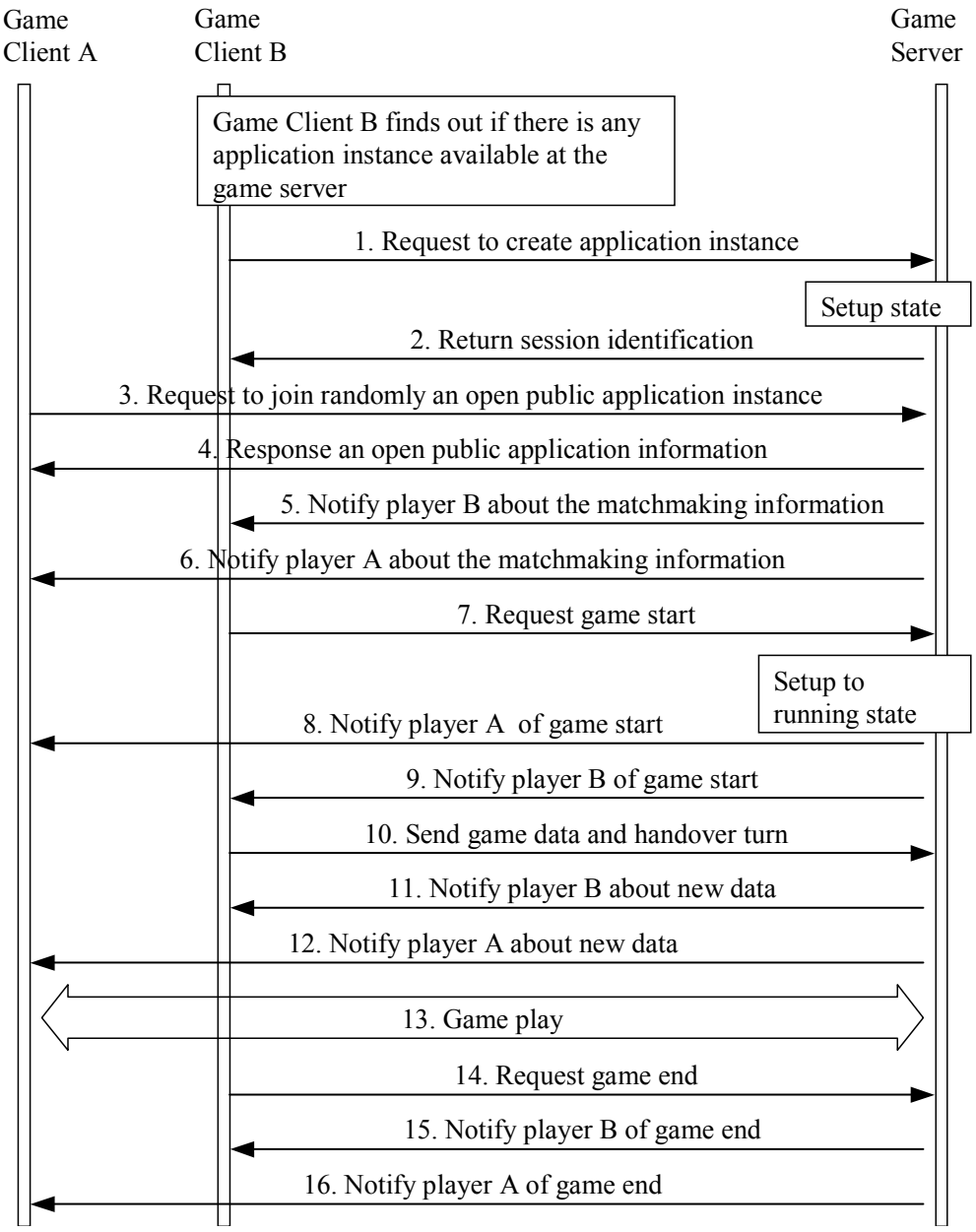

Figure 3: Example of Random Matchmaking

Step 1-2: The game client B requests the game server to create application instance. The game server makes state setup and returns the session identification.

Step 3-4: The game client A requests the game server to join randomly an open public application instance. The game server matches the request to the former application instance and returns the session identification of the instance.

Step 5-6: The game server notifies all the game client of the matchmaking information.

Step 7-9: When the matchmaking is over, the game client B requests the game start. The game server sets the application instance to running state and notifies the clients of game start.

Step 10-13: The game clients send game data. The client may handovers the turn to other client as well if the game is a kind of turn-based. The players enjoy the game.

Step 14-16: When the game is over, the game client B requests the game server to end game. Then the game server notifies all the clients of game end.

4.2.2.3 Highscore

The following Figure 4 illustrates an example of highscore use case. The use case is based on the Use Case 6 Extended Game Experience, Post Data (Saving a score on the server highscore list) [GSRD].

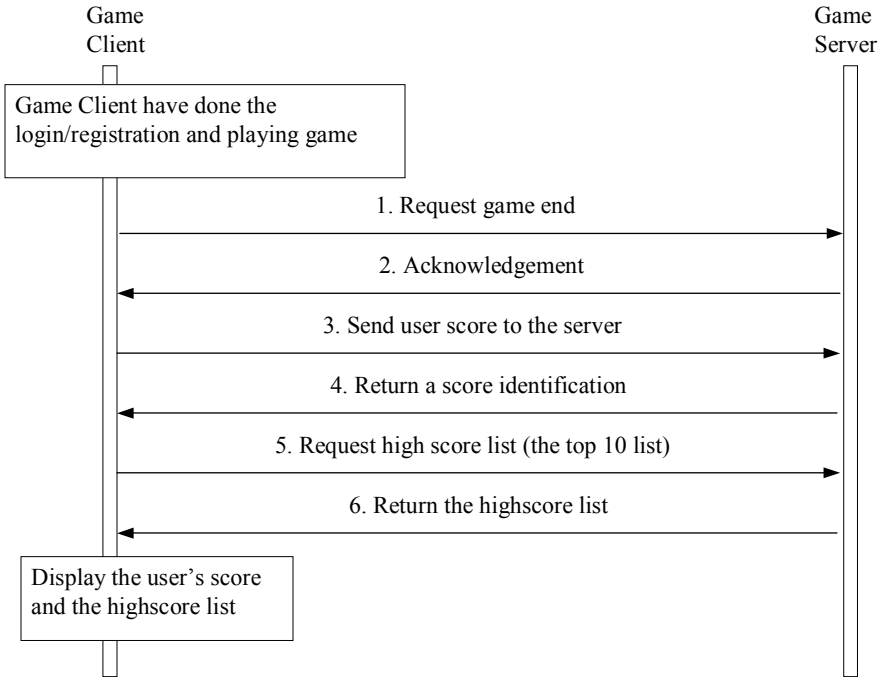

Figure 4: Example of highscore for online game

Step 1-2: When the game is over, the game client request the game server to end of game.

Step 3-4: The game client sends the score of the game playing to the game server. The server returns the identification of score.

Step 5-6: If the game client needs to show the highscore list to the player, it requests the server to return the list. The list is returned, the game client displays the list to the player.

The above Figure 4 is for the online game. As for offline game, where the highscore is uploaded after the game has ended is illustrated by Figure 5:

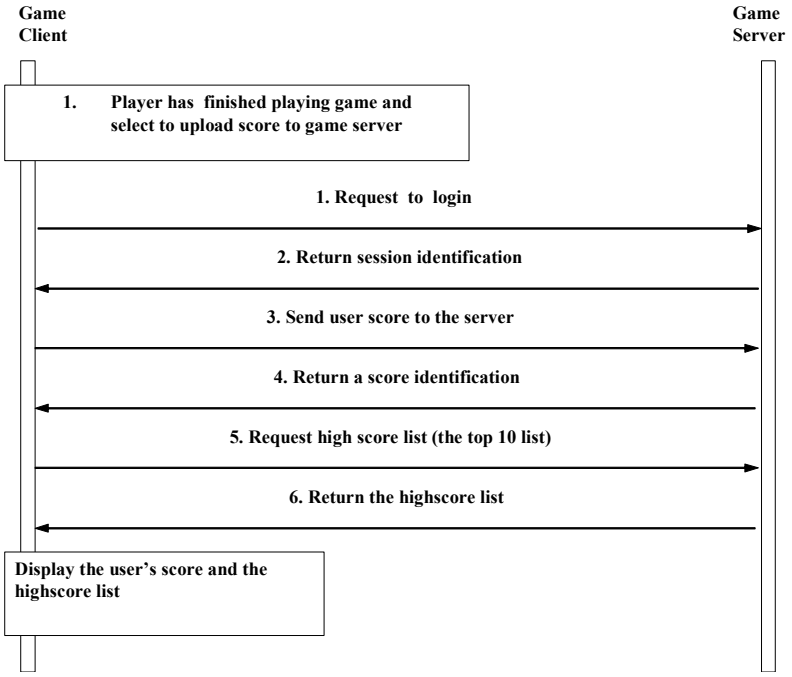

Figure 5: Example of highscore for offline game

Step 1-2: When the player has finished playing game and selects to upload score to game server, he/she requests to login(or registration) the game server.

Step 3-4: The game client sends the score of the game playing to the game server. The server returns the identification of score.

Step 5-6: If the game client needs to show the highscore list to the player, it requests the server to return the list. The list is returned, the game client displays the list to the player.

4.2.3 Real-life Example Scenarios

The scenarios illustrate numbers of the real-life Chinese & Korean mobile game services example in which game server may be used. The game server functionality is not limited to these example scenarios. These scenarios are based on the Use Cases in the OMA Game Service Requirements [GSRD].

4.2.3.1 Turn based game playing

When Mr. Zhang heard that it is easy to play game on the mobile device anywhere and anytime, his interest is inspired to acquire the game services. The following Figure 6 illustrates an example of turn based game playing.

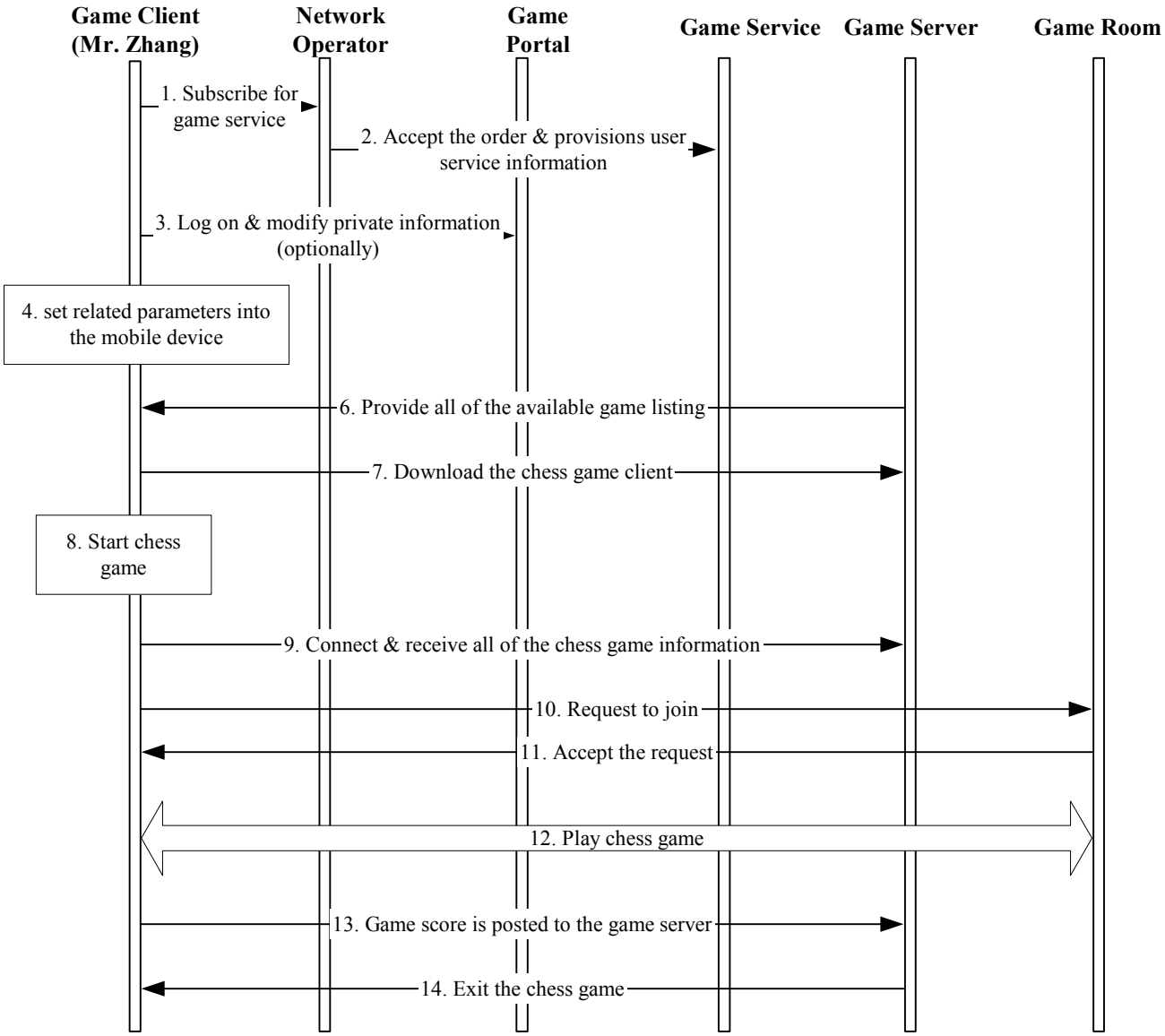

Figure 6: Example of turn based game playing

1. Mr. Zhang subscribes to the network operator for the game service.
2. The network operator accepts the game service order and provisions user service information to the game service.
3. Mr. Zhang logs on to the web portal of the game service by either the PC or mobile phone to modify user private information, such as password. (Optional)
4. He configures game service related parameters into the mobile device, for example the URL of the game server. (Optional)
5. He connects to the game server by the mobile game device, and performs the registration process.

6. After the registration process, the game server provides all of the available game listing.
7. Mr. Zhang is interested in the chess game. He downloads the chess game client into his mobile device, since the chess game client is not available in his device.
8. Upon completion of the download and installation, he starts the chess game.
9. The chess game client connects the game server and receives all the chess game information. (Lobby and other player information)
10. He enters the game lobby and selects one chess game room. He requests to join the chess game instance.
11. The host of the chess game room accepts the request.
12. He plays the chess game with other players on the mobile network.
13. When the chess game is over, the game score is posted to the game server.
14. He exits the chess game.

4.2.3.2 Real-time multiplayer game playing

This section describes a user who wishes to play a real-time multiplayer game. Real-time multiplayer gaming provides the player in real-time to interact with other players, purchase additional games features (i.e. weapons, extra life) based on real-time game points.

Mr. Hong wants to play a 3D-shooter game at this time. The following Figure 7 illustrates the process of real-time multiplayer game playing.

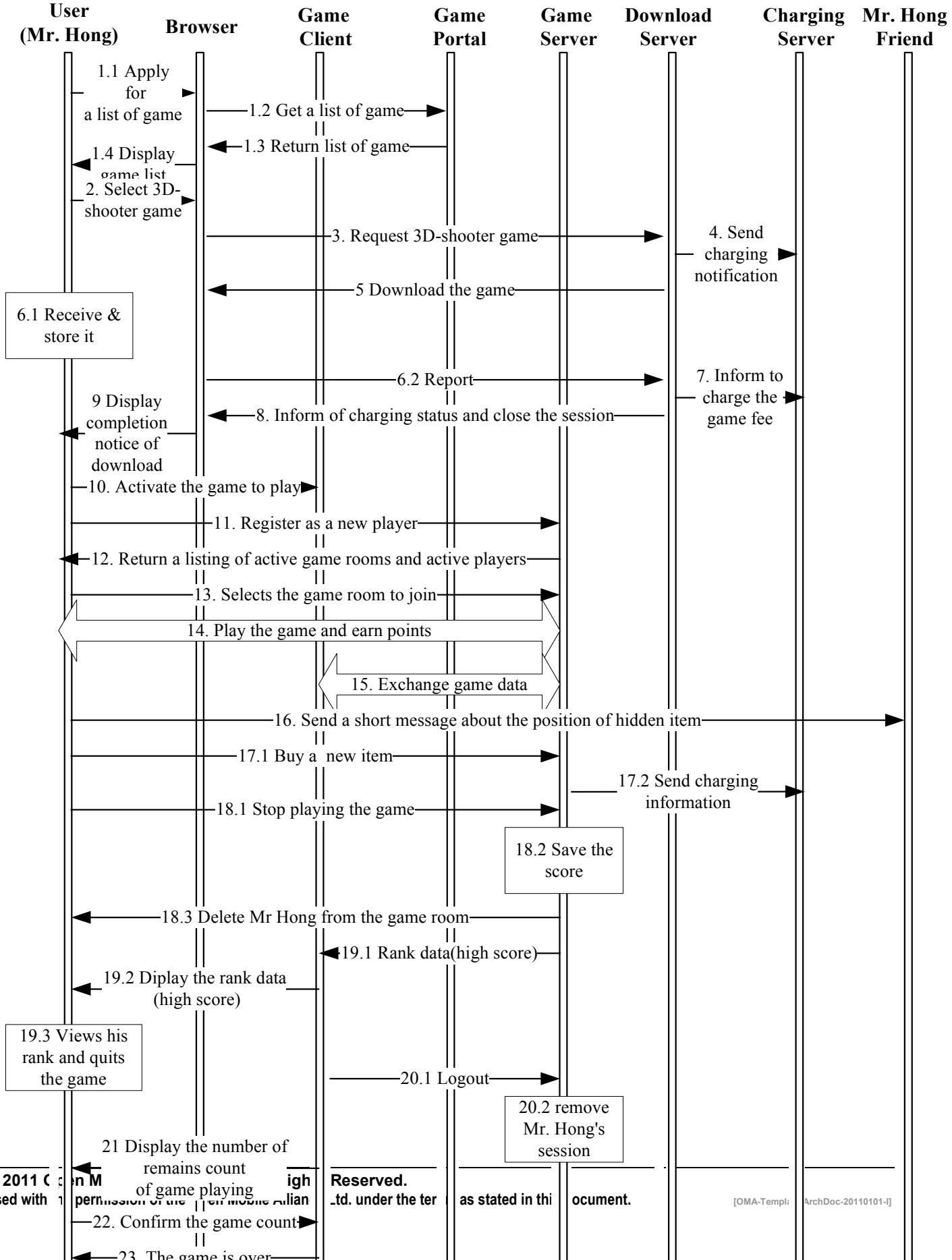

Figure 7: Example of real-time multiplayer game playing

1) Mr. Hong uses a browser to get a list of game. The list may be sorted by popularity ranking.
2) Mr. Hong selects a 3D-shooter game from the list.
3) The browser requesting the 3D-shooter game from the download server.
4) The download server sends a charging notification to the charging system.
5) The download server initiates downloading of the game.
6) The browser receives the game client, stores it, and reports to the download server.
7) Upon receiving the confirmation report from the browser, the download server informs the charging server to bill the game fee.
8) The download server informs the browser of charging status and closes the session between the browser and the download server.
9) The browser displays completion notice of the game download. Mr. Hong confirms it.
10) Mr. Hong activates the game to play.
11) Upon activating the game, Mr. Hong registers to the game server as a new player. The game server creates an account and a user session.
12) Mr. Hong receives a listing of active game rooms and active players from the lobby service of the game server.
13) Mr. Hong selects the game room where he recognizes his friend's nickname. The game server adds Mr. Hong to the selected game room.
14) Mr. Hong plays 3D-shooter game and earns lots of points.
15) During the game, the game client and game server exchange game data dynamically, e.g. stage information, game map, position of players, position of monsters, etc.
16) During the game, Mr. Hong sends a game message to his friend, letting him know the position of hidden item.
17) Mr. Hong buys a new item (e.g. automatic machine gun) from the game shop. The game server sends charging information to the charging server.
18) Mr. Hong stops playing the game and leaves the game room. The game server saves the score. The game server deletes Mr. Hong from the game room.
19) The game client receives the rank data (highscore) from the game server. Mr. Hong views his rank and quits the game.
20) The game client log-out from the game server. The game server removes Mr. Hong session.
21) The game client displays the number of remains count of game playing.
22) Mr. Hong confirms the game count.
23) The game is over.
24) The game application displays the number of remains count of the game playing.

4.2.4 Example of Context Model of the Use Cases

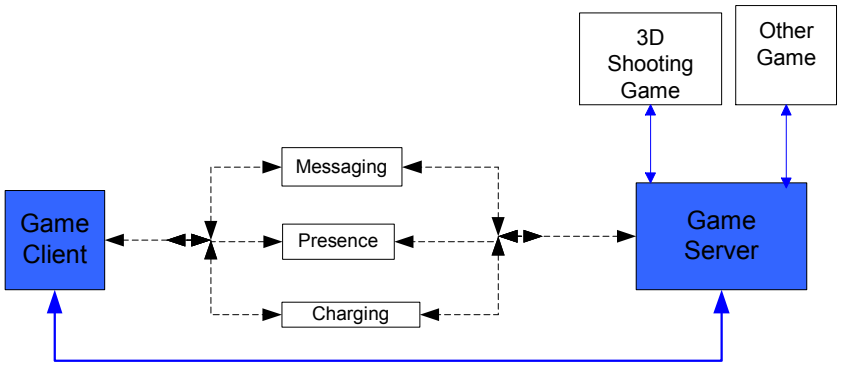

Figure 8: Example of Context Model of the Use Cases

The Figure 8 illustrates context collaboration with other OMA enablers. In Figure 8, the game server provides game centric functionality such as lobby, matchmaking and highscore. Other appropriate OMA enablers, e.g. Messaging, Presence and Charging, would cover the IMS common capabilities.

4.3 Planned Phases

Future releases of the OMA Gaming Architecture might see enhancements in various directions: The OMA GP protocol could be extended to also support user management, more sophisticated high-score and lobby management, event based charging, DRM and single sign-on capability (Liberty Alliance). There could also be integration in-game communication with voice services some time in the future, with instant messaging and SIP.

To standardize the interfaces between service delivery platforms and game server, standards such as Parlay/3GPP OSA could be leveraged as much as possible.

A client library, hiding the complexity of network communication, might be defined, for game developers to use this API when implementing a game client.

5. Context Model (Informative)

5.1 Context Diagram

In the context of mobile game service, OMA Game Service generally consists of game server and IMS service capability [IMSArch]. The game server provides an interface to the game logic, and an interface to the game client, where both client and server store and update the game state according to the game logic. The IMS service capability provides necessary functions to offer a completeness of multi-player online game that is either within or beyond OMA standardization. To help reader, this section and **Figure 9**briefly describes these common capabilities.

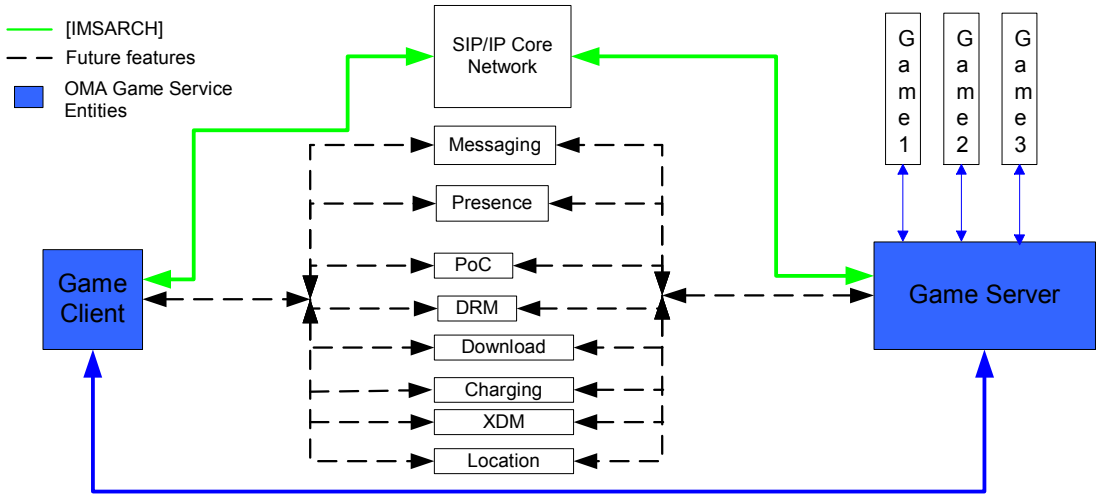

Figure 9: Context Model of OMA Game Architecture

IMS service capability provides many of common capabilities like security, charging, content download, subscriber management, bearer and session control.

The SIP/IP Core Network [IMSArch] is a Session Initiation Protocol (SIP) based IP multimedia infrastructure that provides a fully standardized complete architecture and framework for providing multimedia services. SIP/IP Core Network [IMSArch] provides the architecture for the usage of SIP, and functionalities to support necessary service delivery requirements, such as bearer control, security and charging.

This AD mainly focus on the interaction between game client and game server, and describes the game server function. It is not in the scope of this AD to define all of the IMS service capability. Instead of the AD will be referencing to the necessary specifications, which provide detail of the service capability. Nevertheless, they are briefly described in this document for the completeness of the Game Service. Those are:

The SIP/IP Core provides security, Discovery/Registry, Authentication/Authorization, and Charging. OMA Game Service Enabler can access these SIP/IP Core resources via the I2 interface [IMSArch].

- Presence Service
- PoC Service
- Messaging Service
- DRM

- Download
- Charging
- Location
- XDM (XML Document Management)

The XDM (XML Document Management) Enabler defines a common mechanism that makes user-specific service-related information accessible to the service enablers that need them. Group management is a typical usage among the XDM enablers variety of usages. For example, XDM can be used for a group definition to facilitate session initiation of many individuals to the same game session.

5.2 Context Collaboration Model

At the release of the Game Service Enabler, the Presence, PoC, and Messaging Service Enablers may not be available for Game Service Enabler. Therefore, the Game Service Enabler should utilize the capabilities of the SIP/IP Core as specified in 3GPP [3GPP TS 23.228] and 3GPP2 [3GPP2 X.S0013-002-A].

6. Architectural Model (Informative)

A mobile gaming service is always embedded in a wider technology and business environment. When using any kind of mobile data service, a mobile device has to connect to network delivery equipment of a specific operator. This could include components such as a WAP Gateway, GGSN, SMS-C, MMS-C, Positioning Server, and so on.

Typically, from those network components, all data traffic is forwarded to a service delivery platform, providing generic functionality to a wide range of applications. Such generic functionality can include authentication and authorization, subscriber management; download services, messaging, charging, and many more. Finally, gaming specific traffic gets routed to the game server and the specific games applications, which might be provided by a third party. Figure 10 depicts such a high level set up.

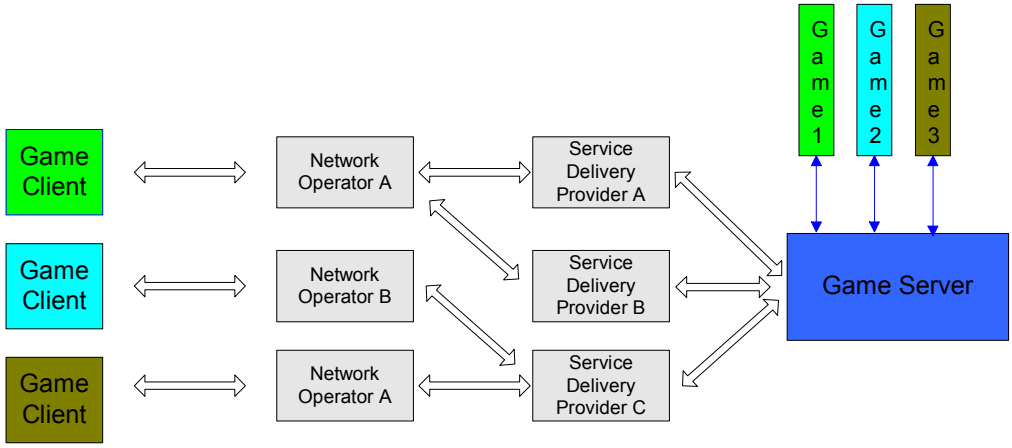

Figure 10: Architecture Overview

Network delivery components can be connected to several service delivery platforms. An example is where a Mobile Virtual Network Operator (MVNO) is using another operators' network equipment. There can also be dedicated service delivery platforms - say for gaming - that can be accessed from multiple networks. And finally the game server and game applications can be dedicated for a service provider or shared across many, to enable cross-operator game play for a large gaming community.

It's important to note, that a game server, as specified by OMA, is not self-contained, but instead relies on the functionality provided by both the network delivery components, as well as the service delivery platform. Subscriber management, content download, messaging, or charging are good examples: this functionality can be leveraged from the service delivery platform and hence is out of scope for the OMA Gaming Service.

The OMA Gaming Service can be regarded as generic infrastructure on the server side providing common services to a range of game applications.

6.1 Architecture Diagram

The architecture for the OMA Game Service Enabler and other OMA enablers are described in Figure 11.

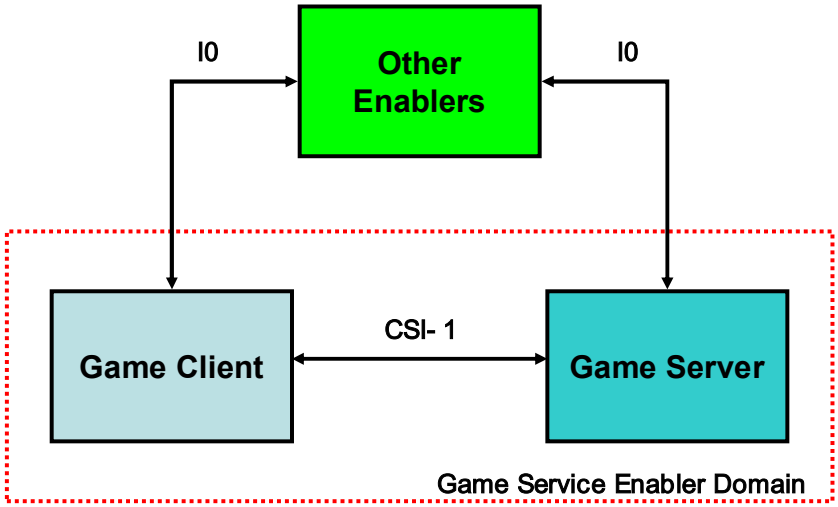

Figure 11: Architecture diagram of Game Service Enabler

The CSI-1 describes the interface between the Game Client and the Game Server in the Game Service Enabler Domain. This interface is the Game Client-Server Interface.

The I0 in Figure 11 indicates the interface between the Game Service Enabler and other OMA enablers. These interfaces are defined by the other OMA enablers.

6.2 OMA Game Services CSI Enabler Release 1.0

The OMA Game Services CSI Enabler Release Version 1.0 contains the Client/Server Interface version 1.0 [ClientServerInterface].

1. The OMA Client/Server Interface Specification [ClientServerInterface] specifies the communication between a game client on a mobile device and a game server, in which the server side part of the game is hosted. Both, the game client and the game server themselves are not specificied, but considered as black boxes. The defined Interface can run on top of various transport layers, such as e.g. HTTP, SOAP, or UDP. This layer depends on the underlying network and device capabilities.

The [GP10] and the GS CSi 1.0 are two independent enablers.

The event triggering mechanism in [GP10] is not aligned with the GS CSI 1.0 event mechanism.

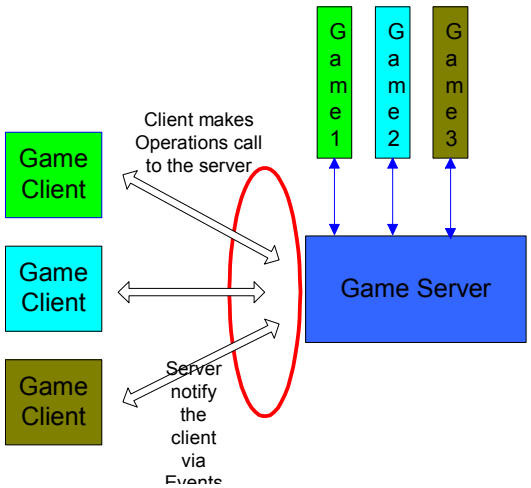

Figure 12: Client/Server Interface

As can be seen in Figure 12, the OMA Client/Server Interface [ClientServerInterface] exposes a set of API interfaces, including the following operations:

- Login and Registration
- Game Persistence
- Game Creation and Matchmaking
- Game Play
- Messaging
- Highscore
- Miscellaneous

Besides those interfaces, there are events defined, which can be polled from the mobile device. Such events can be that a new player has joined a game or someone has left, that a turn is handed over, that a text message has been received, or similar.

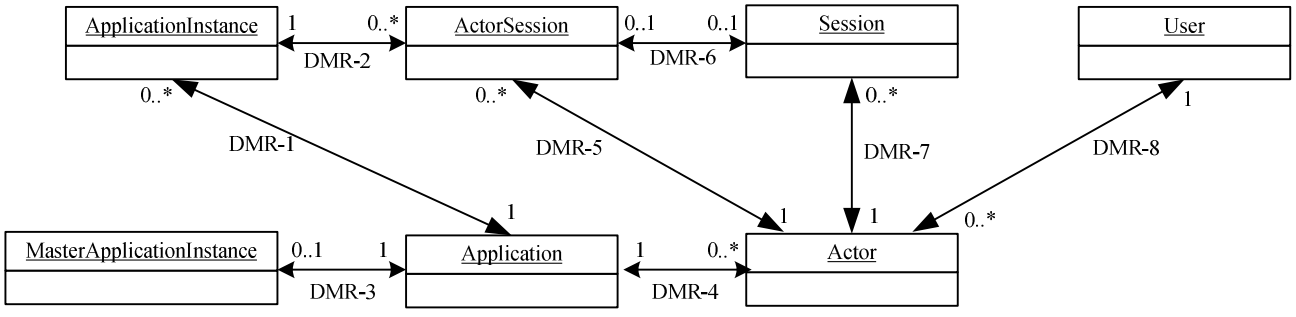

Figure 13: Domain Model of an OMA Client/Server Interface version 1.0

The domain model has been slightly enhanced in the Client/Server Interface version 1.0, as shown in the Figure 13. The DMR (Domain Model Relationship) arrows in Figure 13 are not to be confused with the CSI 1.0 interfaces!

The only changes from the OMA GP 1.0 domain model are the introduction of the new object "*Session*", representing the communication between an Actor and the active ActorSession; some label names are giving for the convience of mentioning the relationship between two objects:

DMR-1 represents the relationship between Application and ApplicationInstance. It is one-to-many relationship.

DMR-2 represents the relationship between ApplicationInstance and ActorSession. It is one-to-many relationship.

DMR-3 represents the relationship between Application and MasterApplicationInstance. It is one-to-many relationship.

DMR-4 represents the relationship between Application and Actor. It is one-to-many relationship.

DMR-5 represents the relationship between Actor and ActorSession. It is one-to-many relationship.

DMR-6 represents the relationship between ActorSession and Session. It is one-to-one relationship.

DMR-7 represents the relationship between Actor and Session. It is one-to-many relationship.

DMR-8 represents the relationship between User and Actor. It is one-to-many relationship.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-AD-Game-Services-Architecture-V1_0-20110329-A	29 Mar 2011	Status changed to Approved by TP: OMA-TP-2011-0094-INP_GS_CSI_V1_0_ERP_for_Final_Approval

Appendix B. Enabler Release 1

OMA's Game Platform 1.0 [GP10] specification includes architecture, an object model, and a set of APIs [GP10JDOC], which game developers can use, to run their game application on an OMA 1.0 compliant platform.

Emphasis of the release 1.0 work has been on *games portability* (running a game application on a number of server side games platforms) and *games interoperability* (enabling game applications executing in different environments, to exchange information and share common services) of the server side game logic.

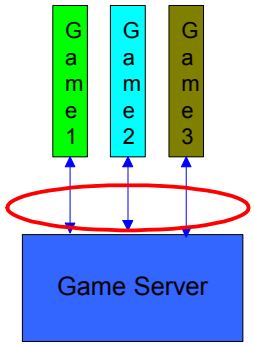

Figure 14: Server interface for Game Applications

In a nutshell, GP 1.0 is defining the interface of an OMA gaming server, which game applications can use in an environment, where both the game server and the game application are executed on the server side. APIs have been specified and are addressing the following areas:

- Session management
- Connectivity
- Metering
- Scores and Competition Management
- Logging.
- Timers

Despite the fact that OMA standards are language neutral in nature, a set of Java APIs is provided as an example. From this, a set of interfaces can be derived for C++, C# or other programming languages.

Besides this set of interfaces, a domain model has been defined capturing the essence of an OMA Gaming Platform 1.0.

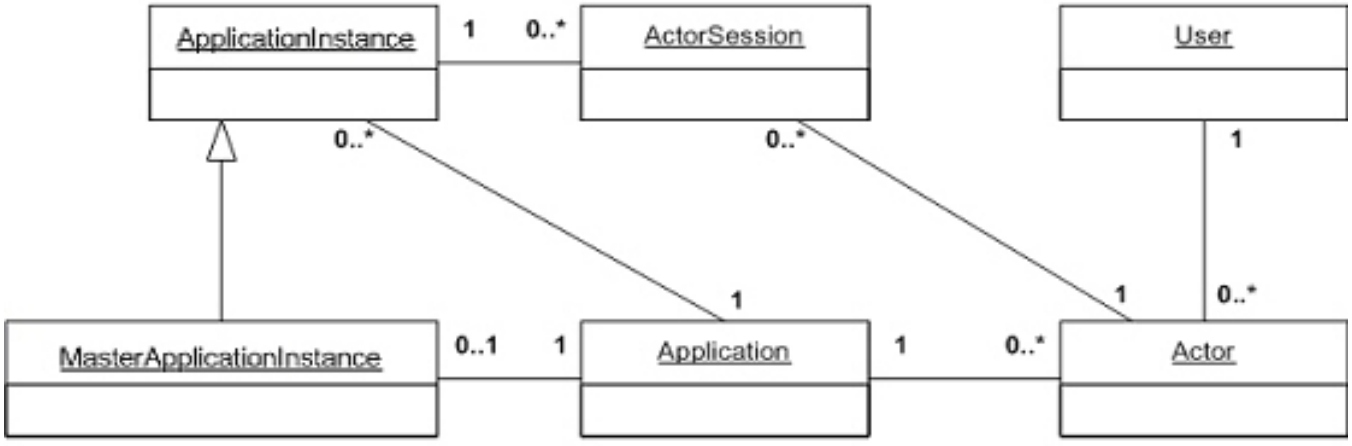

Figure 15: Domain Model of an OMA Gaming Platform 1.0

The semantics of this model is as follows: A User is a phone subscriber who has a cellular account and may access the gaming server. *Users* are typically identified by their MSISDN or their user name and password. For the gaming server, it is assumed that each user has an internal unique UserID.

A user can play multiple games at the same time or maybe even act as several users in the same application. This concept is called an *Actor*. An *Actor* represents shared information between all *ActorSessions* in the same *Application*.

An *Application* is the installed code or logic of a game. The *ApplicationInstance* is the actual game played; it is the running instance of an *Application*. When the user selects an *Application* to play, an instance of the *ActorSession* class is created for this user.

An *ActorSession* object represents a specific user in the context of a particular *ApplicationInstance*. Each user can be present in multiple applications at the same time, and thus be associated with several *ActorSession* objects. There is a one-to-many relationship between the *Actor* and *ActorSession*. The relation between *ActorSession* and *ApplicationInstance* may similarly be many to one.

The *MasterApplicationInstance* represents a specific application instance, which is used to manage events and information, shared between *ApplicationInstances*. This object can be used for special management events, e.g. declarative timers.