Unleash Your Choices

How TPI Enables More Game Content for an Operator

Unleash Your Choices: How TPI Enables More Game Content for an Operator

Released 2019/10/16, by Gaming Standards Association® (GSA®).

Patents and Intellectual Property

NOTE: The user's attention is called to the possibility that compliance with this [standard/specification] may require use of an invention covered by patent rights. By publication of this [standard/specification], GSA takes no position with respect to the validity of any such patent rights or their impact on this [standard/specification]. Similarly, GSA takes no position with respect to the terms or conditions under which such rights may be made available from the holder of any such rights. Contact GSA for further information.

Trademarks and Copyright

GSA Contact Information

E-mail: sec@gamingstandards.com

WWW: http://www.gamingstandards.com

Table of Contents

About This Document

This document provides a high-level description of the TPI standard. It is designed to provide the reader with a conversational overview of TPI. This document is not intended to be a reference or technical document detailing all functionality. Refer to the appropriate standards documentation for complete information. Contact GSA for information on training materials, online classes and on-site classes.

1.1 Acknowledgements

The Gaming Standards Association would like to express its appreciation to all members of the different committees, past and present, for their contribution and dedication to the creation of the standards upon which this document is based.

1.1.1 Document Formatting Conventions and Organization

Blue text indicates an internal link or external hyperlink to a URL.

Bold (other than in headings) or underlined text is used for emphasis, unless specifically indicated otherwise.

Italicized text (other than in headings and document titles) is used for terms being defined.

`Courier New` font is used to indicate protocol components and their values.

CHAPTER

1

Introduction

1.1 What is Third-Party Game Interface?

Third-Party Game Interface, or TPI, provides a standard set of commands that online games can use to communicate with gaming platforms and jackpot controllers. TPI handles all the integration needs of online games, gaming platforms, and jackpot controllers, providing all the communications needs for operating online games.

TPI is designed specifically for online gaming. The TPI specification describes a standardized interface between iGaming Platforms, Remote Game Servers, and Progressive Jackpot Controllers. TPI contains messages for launching games, recording monetary transactions, posting progressive contributions, awarding progressive jackpots, reconciling interrupted games, and more.

TPI fully supports online gaming operations that service multiple jurisdictions, allowing the activity associated with each jurisdiction to be isolated and reported. The specification is based on JSON, HTTP/REST, and WebSocket technology.

With TPI you can:

- determine available games on an RGS
- configure those games for use with an iGaming platform
- create betting configurations for games

- create free-spin configurations for games
- construct URLs and launch games on an RGS
- start and end game sessions on an RGS
- finish suspended games
- communicate state information between game content and an iGaming console
- report wagers and wins from game play
- report contributions to jackpot controllers
- manage jackpot awards - for standard or mystery jackpots

TPI does not include other messages that may be necessary to operate an iGaming website, such as

Note: These types of messages are beyond the scope of the TPI specification.

- messages for logging onto the website
- transferring funds to the website
- verifying player identification
- establishing geo-location

1.2 TPI Terminology

Before describing the various services in TPI and how they interact, some terminology must be defined.

Table 1.1 TPI Terminology

Term	Description
Game Cycle	A *Game Cycle* is an instance of a game. A game cycle usually starts when the player places a bet on a game. The game cycle ends once the outcomes of all bets are known. It can remain open for an extended period of time. This frequently is the case with sports betting. Multiple subsequent bets can be made during a game cycle.
Game Cycle Group	A *Game Cycle Group* links together game cycles that share the same outcomes. (example: multi-player roulette or poker). Multiple game cycles will be dependent on the same set of outcomes. This feature is only used when appropriate.
Game Launch URL	A *Game Launch URL* is used by TPI to pass information to the RGS for use in starting game sessions and selecting betting configurations for individual players. The game launch URL is the first step towards establishing a game session for a player.
Game List	The game list provides information about the set of games on the RGS. It includes the static properties for the games (properties that cannot be changed), the protocol-required properties (properties that must be configured when a betting configuration or free-spin configuration is created) and the game-specific properties (additional properties defined by the supplier).
Game Session	A *Game Session* is established when a game is successfully launched on the RGS. Multiple game sessions can be launched during a player session. They could be for the same game or for different games, possibly on a different RGS. Activity related to game sessions is reported by the RGS to the iGP through TPI.
iGamingConsole	The *iGamingConsole* (iGC) allows the player to access the administrative services on the iGP and launch games.
iGamingPlatform	The *iGamingPlatform* (iGP) provides administrative services for the iGaming operation.

Table 1.1 TPI Terminology

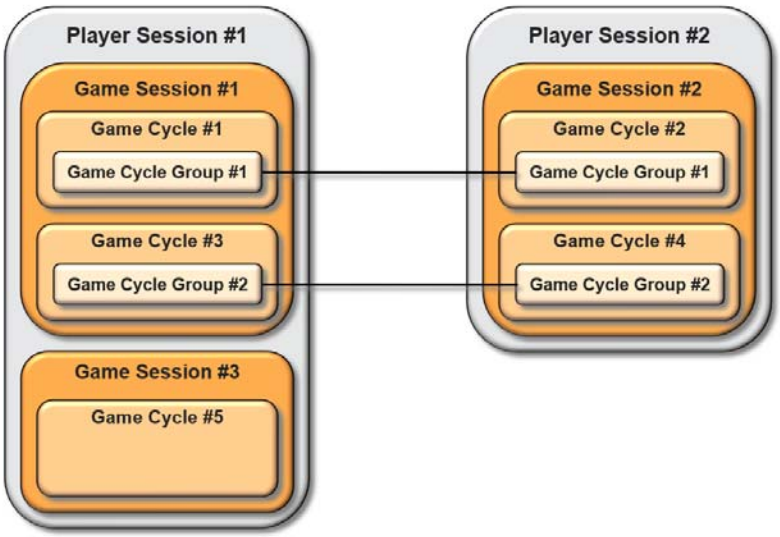
Term	Description
Jackpot Controller	A *Jackpot Controller* is a specific group of jackpots managed by a PJS. The PJS may manage more than one jackpot controller – more than one group of jackpots.
Jackpot Controller Type	The *Jackpot Controller Type* indicates whether a jackpot controller supports standard or mystery jackpots.
Jackpot Level	A *Jackpot Level* is a specific jackpot, or pool of money, within a jackpot controller. There can be many jackpot levels within a jackpot controller.
Player Session	A *Player Session* is established when the player successfully logs onto the iGP. The player session ends when the player logs off or is disconnected from the iGP. Activity related to the player session is not reported through TPI. It is iGP-specific. However, information about the player session – such as the jurisdiction in which the player is located – is communicated to the RGS.
Progressive Jackpot Service	A *Progressive Jackpot Service* (PJS) is a specific endpoint that supports jackpot functionality through TPI. The PJS provides the jackpot services to the iGaming operation. It could be provided by the game supplier or an independent systems supplier.
Remote Game Service	The *Remote Game Service* (RGS) provides the game services to the iGaming operation. The RGS picks an appropriate betting configuration based on parameters in the game launch URL.
Remote Game Console	The *Remote Game Console* (RGC) allows the player to play the games offered by the RGS.

1.2.1 Player Sessions and Game Sessions

The image below illustrates the relationships between player sessions, game sessions, game cycles, and game cycle groups.

There are two player sessions: player session #1 and player session #2.

Game sessions for a multi-player roulette game are launched from both player sessions. The players are both playing the same game. This is shown as game cycle group #2.



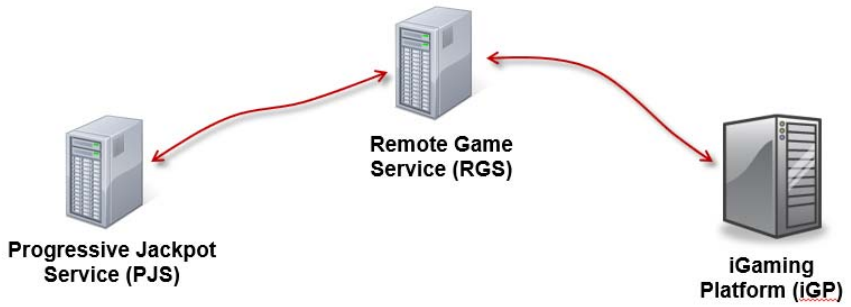
Two rounds, or game cycles, are played. Game cycle groups are used to identify the rounds: game cycle group #1 and game cycle group #2.

After the two rounds have been completed, both players exit the roulette game. The second player — player session #2, exits the player session.

The first player, player session #1, plays another game — a single player spinning reel game — a new game session is launched. This is shown as game session #3. The player plays one game — one game cycle — and then exits the game and the player session.

1.3 TPI Topology

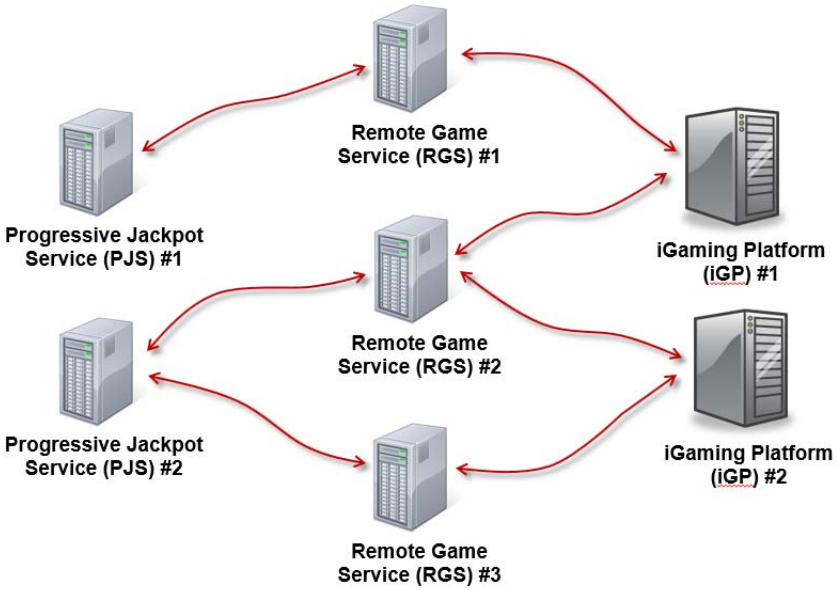
TPI topology consists of three end-points: The iGaming Platform (iGP), Remote Game Service (RGS), and Progressive Jackpot Service (PJS). The image below displays TPI topology in its basic form. In this image, the three end-points are shown as separate, distinct servers.



Configurations can be much more complex. Any of the three end-points can be comprised of multiple physical servers. Or a single physical server can host more than one service – for example, an RGS and a PJS can reside on the same physical machine.

The image below shows another more complex configuration.

Benefit: TPI's flexible framework can be adopted to many operational scenarios.



Configurations are not limited to a single iGP, or a single RGS, or a single PJS. Configurations involving multiple iGaming platforms, remote game

services, and progressive jackpot services are possible. Each server is expected to communicate with other servers in a manner appropriate to their roles.

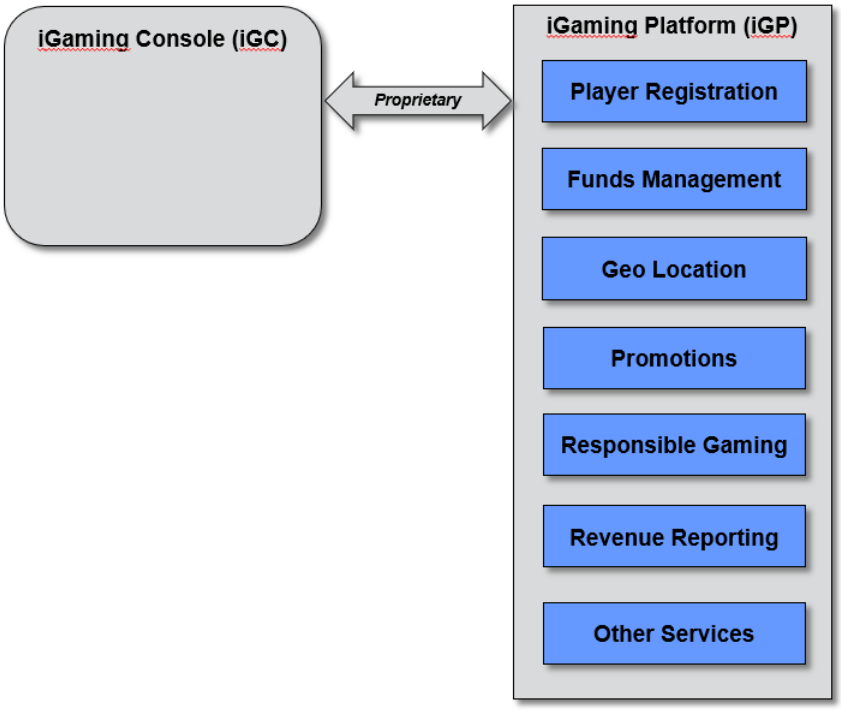
1.3.1 iGaming Platform

In TPI, the iGaming Platform is composed of two distinct parts: the iGaming Console (iGC) and the iGaming Platform (iGP) itself.

The iGC is the interface the player uses to view content on their PC, mobile phone, or tablet. It is the portal through which the player communicates with the iGP. The iGC also allows the player to launch and play games and access administrative services on the iGP.

The iGP provides administrative services for the iGaming operation. This includes services such as player registration, identity verification, funds management, geo location, promotions, responsible gaming, revenue reporting, and other related services.

TPI does not address the communications between the iGP and the iGC. The supplier of the iGP and iGC can use whatever method it feels is appropriate. (shown as *Proprietary* in the image below)



1.3.2 Remote Game Service

The RGS is composed of two distinct parts: the remote game content (RGC) and the remote game service (RGS) itself.

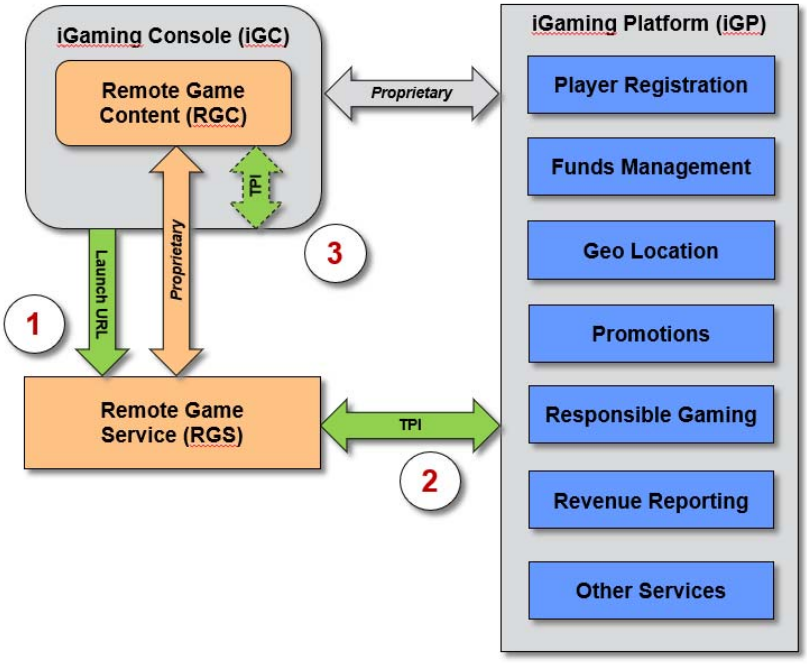
The RGC allows the player to play the games offered by the RGS. The RGC is the content the player sees while playing a game. Sometimes this content will be displayed in a specific area within the iGC. Other times, it will appear in a separate independent window on the player's device; or, it will completely replace the iGC.

The RGS manages and controls the game content. It provides game services to the iGaming operation. The RGS selects the appropriate betting configuration for the player and determines the game outcomes, reporting the associated wagers and wins to the iGP.

TPI specifies the information to be included in the game launch URL. TPI also specifies how the RGS should report wagers and wins to the iGP.

TPI also specifies what data should be communicated between the RGC and the iGC – such as language and player account balances. Referring to the image below, this is shown as the arrow with dotted lines (item #3). The lines are dotted as a way of indicating that TPI only specifies what data should be communicated, not how it should be communicated.

Note: The launch URL is used by the iGaming console to launch a game on the RGS (item 1 in the illustration).



TPI can also be used by the iGP to discover the games available on the RGS and configure those games for use with the iGP. This involves creat-

ing specific betting configurations and free-spin configurations for the brands and skins supported by the iGP.

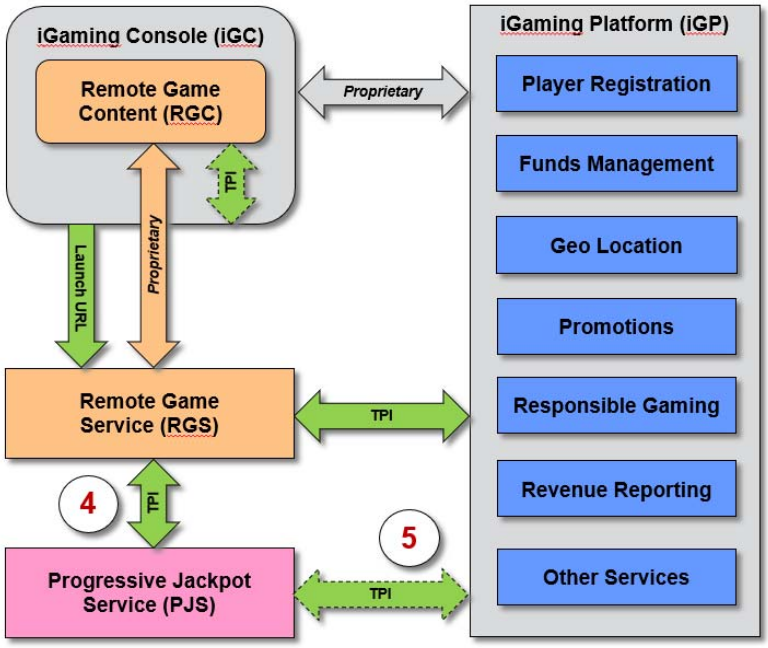
TPI does not address the communications between the RGS and the game content. This is considered proprietary. The supplier of the RGS and game content can use whatever method it feels is appropriate.

1.3.3 Progressive Jackpot Service

The PJS is responsible for managing jackpots. Standard progressive jackpots and mystery jackpots are both supported. A single PJS can manage jackpots for multiple RGS's and multiple games within those services.

TPI specifies how contributions are reported from the games to the jackpots and how jackpot awards are managed.

TPI can also be used to send jackpot information to the iGP – for example, the iGaming platform may want to display current jackpot values to the players. This is shown in the image below as item #5. The lines are dotted to point out that communications between the jackpot service and iGaming platform is optional – it is specified within the protocol, but is only required when operational or jurisdictional requirements so dictate.



All commands in TPI are constructed using JavaScript Object Notation (JSON). Two message transport options can be used; JSON over WebSockets, or JSON over HTTP. For more information see "*Command Structure & Game Launch URL*" on page 19.

CHAPTER

2

Using TPI

2.1 Introduction

This chapter provides a high-level review of typical TPI gaming activity. This includes starting, playing, and ending games.

2.2 Playing A Game

A player decides to play a game on a PC, tablet, or mobile device. First, the player logs into the operator's website. Once logged in, the player selects a game to play from the iGC. At this point, TPI comes into the picture.

The iGC sends a game launch URL with the information about the player and selected game to the RGS.

Once the RGS has this information, it verifies the information with the iGP (using the `playerSession` command). Once acknowledged, the RGS starts the game session.

The player may invoke multiple game sessions and multiple games. A game session typically ends when a player decides to exit the game.

2.3 Game Launch URL

The game launch URL is at the heart of establishing a game session for a player on an RGS. The game launch URL contains a standard set of parameters. It may also contain game-specific parameters. These parameters can be defined on a game-by-game basis by the supplier. The game launch URL is formed by appending the standard launch parameters and the game-specific launch parameters to the base URL of the game.

Based on the launch parameters, the RGS picks an appropriate betting configuration for the game session.

The table below contains the standard set of launch parameters.

Table 2.1 Game Launch URL Parameters

Parameters	Description
`igpId`	Unique iGaming Platform Identifier.
`rgsId`	Unique Remote Game Service Identifier.
`playerId`	Unique Player Identifier.
`accountId`	Unique Player Account Identifier.
`brandId`	Primary identifier for a specific ‘look and feel’ offered by the iGaming operator.
`skinId`	Secondary identifier.
`channelType`	Identifies the distribution channel through which the game will be played.
`presentType`	Identifies the format in which the game will be rendered. For example: HTML5, FLASH10, or an executable.
`localeCode`	Identifies the language and country variant in which the game should be presented.
`currencyCode`	Identifies the currency in which the game will be played.
`gameId`	Identifies a specific game on the RGS.
`secureToken`	An authorization key which uniquely identifies the game session launched by the iGP.

For more information, see "*Command Structure & Game Launch URL*" on page 19.

2.3.1 Betting Configuration

Before a game can be launched, at least one betting configuration must be created for that game. This can be done locally at the EGM or it can be done by the iGP using commands within TPI.

Note: When a game is launched, the betting configuration used must be appropriate for the launch parameters.

Various protocol-related properties are specified when a betting configuration is created. These properties are defined in the protocol. Some, such as brand and skin, appear in the game launch URL and, thus, have to do with the selection of a betting configuration. Others, such as paytable and base currency, have to do with the behavior of the game after it is launched.

Game-specific properties can also be specified when a betting configuration is created. These properties are defined by the supplier of the game – they are unique to the specific game – things like game speed or minimum bet. This information is communicated through the protocol, but is not part of the protocol itself.

For more information, see *"Betting Configuration States*" on page 37.

2.4 Player Sessions and Game Sessions

A player session starts when the player logs onto the iGP. A game session starts when a game is successfully launched on an RGS. During the course of the player session, multiple game sessions may be launched. These game sessions might be for the same game or for different games, and might possibly reside on a different RGS.

Once the iGP has verified the parameters of the game launch URL, the RGS can select a betting configuration for the game, start a game session, and then launch the game. If verification fails, a game session should not be started.

2.5 Playing Games

Game play is a three-step process.

- The game cycle starts.
- Wagers and wins associated with the game cycle are reported as monetary transactions — the wagers and wins can be spread across multiple monetary transactions.
- The game cycle ends.

A game cycle is a subset of a game session. A game session must be open to start a game (game cycle). A game session does not need to be open to end a game cycle.

For more information, see *"Game Play Operations*" on page 45.

2.5.1 Ending a Game Session

A game session typically ends when the player leaves the game either voluntarily by existing the game or involuntarily because of a browser failure, network interruption, etc. Game sessions may also end for operational or jurisdictional reasons.

2.5.2 Closing a Game Session

Typically, game sessions are ended by the RGS. However, the iGP can also force the RGS to end a game session and, if desired, start a new one. For example, the iGP might want to force a game session to end when a player moves into another jurisdiction where online gaming is not allowed.

2.6 Jackpots

TPI provides a full range of functionality for managing jackpots. It includes commands for:

- discovering the jackpots available on a PJS
- discovering the links between games on an RGS and the jackpots on a PJS
- reporting contributions from an RGS to the jackpots on a PJS, and
- managing the award of jackpots by a PJS.

Standard jackpot wins are determined by the RGS, usually based on the player hitting a specific winning combination in a game.

Mystery jackpot wins are determined by the PJS, usually based on the jackpot reaching a randomly selected value. Other triggers for mystery jackpots are also possible.

The image below illustrates the relationships between progressive jackpot services, jackpot controllers, and jackpot levels.

In this example, the progressive jackpot service contains three controllers: jackpot controllers 1, 2, and 3. Jackpot controller 1 is a standard progressive controller; jackpot controllers 2 and 3 are mystery controllers.

Jackpot controller 1 has five jackpot levels; controller 2 has two levels; and controller 3 has one level.

There are no restrictions on how many controllers can be supported by a jackpot service or how many levels are within a controller.

2.6.1 Jackpot Configuration

Jackpot controllers and jackpot levels are configured on the PJS. The RGS and iGP can read, but not change, the jackpot configuration information. TPI does not contain commands for configuring jackpots, only for reporting the configurations.

Jackpot configuration information includes parameters such as the contribution rate, jackpot reset value, etc.

The links between games and jackpot levels are configured on the RGS. Like with the jackpot configuration information, the PJS and iGP can read the jackpot link information, but cannot change it. TPI does not contain commands for configuring jackpot links, only for reporting the links.

CHAPTER

3

Command Structure & Game Launch URL

3.1 TPI Command Structure

All commands in TPI are constructed using JavaScript Object Notation (JSON). Schemas supporting TPI functionality are included in the protocol distribution package.

Commands are always defined in request-response pairs. There is always a well-defined response for each request.

3.1.1 Request Properties

All requests contain a standard set of properties. Requests always include the identifiers for the two end-points exchanging information — the iGaming Platform Identifier (`igpId`), the Remote Game Service Identifier (`rgsId`), or the Progressive Jackpot Service Identifier (`pjsId`). Two out of three end-points must always be included in a request.

Requests must include exactly one command.

Table 3.1 TPI Command Request Properties

Property	Description
`igpId`	iGaming Platform Identifier.
`rgsId`	Remote Game Service Identifier.
`pjsId`	Progressive Jackpot Service Identifier.
`requestId`	Request Identifier - used to match responses to requests.
`Command`	Command name - identifies the command being sent.
`data`	A JSON object containing the properties of the command.

3.1.2 Response Properties

As with requests, responses contain a standard set of properties. Responses always include the identifiers of the two end-points and the Request Identifier from the request.

Responses must include exactly one command or an error code.

Table 3.2 TPI Response Properties

Property	Description
`igpId`	iGaming Platform Identifier.
`rgsId`	Remote Game Service Identifier.
`pjsId`	Progressive Jackpot Service Identifier.
`requestId`	Request Identifier.
`Command`	Command name - identifies the response being sent.
`data`	An object containing the application-level response.
`errorCode`	Error code. (Only included if there is an error.)
`errorMsg`	Error message. (Only included if there is an error.)

3.2 Message Transport

TPI can work with two transport scenarios:

- JSON over WebSockets
- JSON over HTTP

3.2.1 JSON Over WebSockets

When JSON over WebSockets is used, the RGS is responsible for establishing a WebSocket connection to the iGP and, if necessary, to the PJS. The PJS is responsible for establishing a connection to the iGP, if necessary.

Both end-points can use the same WebSocket connection to initiate requests. Only one connection is needed for full two-way communications.

3.2.2 JSON Over HTTP

When JSON over HTTP is used, the end-point that initiates a request is responsible for establishing a connection to the recipient. Thus, if an RGS needs to send a request to an iGP, the RGS needs to establish an HTTP connection to the iGP. If the iGP needs to send a request to the RGS, the iGP needs to establish an HTTP connection to the RGS.

Two connections are needed for full two-way communications.

3.2.3 Persistent Connections

In both cases, persistent connections are required. Unless an end-point no longer needs a connection, the end-point is expected to keep the connection alive for at least five minutes.

3.3 Connection URI

The URI used to make a connection must contain the identifiers of the two end-points. For example, if an RGS is making a connection to an iGP, the URI must include both the RGS Identifier (`rgsId`) and the iGP Identifier (`igpId`).

The end-point to which the connection is being established is referred to as the service host. The service host is expected to validate the two identifiers and, if the identifiers are invalid, return HTTP status code 400 "Bad Request".

For example, if an RGS is establishing a connection to an iGP, the iGP is expected to validate the identifiers. If the `igpId` or the `rgsId` is invalid, the iGP is expected to return HTTP status code 400.

Both end-points are expected to verify that the identifiers used in requests match the identifiers specified when the connection was established. If they do not match, an appropriate error code should be returned in the response.

3.4 Game Launch URL

The game launch URL is used by TPI to pass information to the RGS for use in selecting betting configurations for individual players and launching games. The game launch URL is the first step towards establishing a game session for a player. The information which is passed along is done so with parameters. Refer to the table below for a complete list of all the parameters.

Table 3.3 Game Launch URL Parameters

Parameters	Description
`igpId`	Unique iGaming Platform Identifier.
`rgsId`	Unique Remote Game Service Identifier.
`playerId`	Unique Player Identifier.
`accountId`	Unique Player Account Identifier.
`brandId`	Primary identifier for a specific ‘look and feel’ offered by the iGaming operator.
`skinId`	Secondary identifier.
`channelType`	Identifies the distribution channel through which the game will be played.
`presentType`	Identifies the format in which the game will be rendered. For example: HTML5, FLASH10, or an executable.
`localeCode`	Identifies the language and country variant in which the game should be presented.
`currencyCode`	Identifies the currency in which the game will be played.
`gameId`	Identifies a specific game on the RGS.
`secureToken`	An authorization key which uniquely identifies the game session launched by the iGP.

3.5 iGP, PJS, and RGS Identifiers

Each end-point in an iGaming network is assigned a unique identifier. There are three possible end-point identifiers; the *iGP Identifier*, *RGS Identifier*, and *PJS Identifier.*

Tip: The URLs of the end-points are not exposed through TPI. They must be configured into the systems.

The iGaming network administrator is responsible for assigning the identifiers and assuring that they are unique. Systems should include a facility for registering the identifiers, as well as their associated URLs, so that the identifiers can be verified, and connections established.

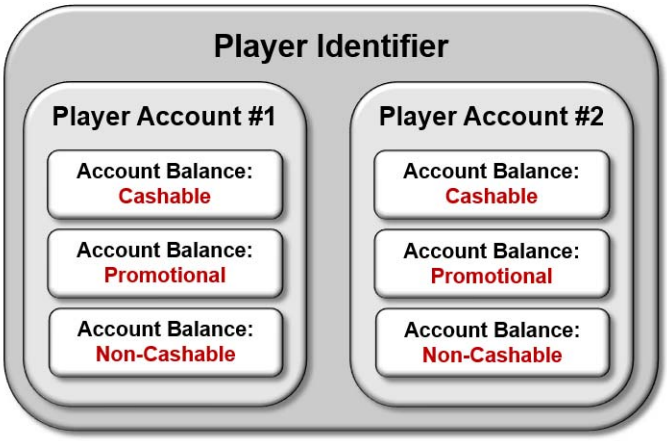
3.6 Player Identifiers and Player Account Identifiers

Each player is assigned a unique *Player Identifier* by the iGP. There may be multiple player accounts associated with a single player identifier. Each account is also assigned a unique *Player Account Identifier*.

Each player account is denominated in a single currency.

A player may have an account based in Dollars and another account based in Euros, but an individual account may only have one currency. Within a single player account, the player may have multiple account balances: cashable, promotional, or non-cashable.

The image below illustrates the relationships between player identifiers, player accounts and the three different account balance types. It is possible to have multiple account balances within an account with the same account balance type – for example, multiple non-cashable balances.



3.6.1 Blocked Accounts

One or more account balances within a player account may be blocked. Blocked funds cannot be used for gaming purposes. Blocked funds cannot be wagered.

3.7 Brand Identifier & Skin Identifier

The *brand identifier* and *skin identifier* identify a specific "look and feel" offered by the iGaming operator. Typically, the brand is the primary identifier and the skin is the secondary identifier.

An iGP may support multiple brands; it may also support multiple skins per brand. These may be for the same iGaming operator or for different operators.

Similarly, an RGS may support multiple brands and skins. Some games on the RGS may be restricted to specific brands – they may contain brand-specific symbols. Others may be available to all brands and skins.

The iGaming operator, in consultation with the iGP and RGS suppliers, will choose brand and skin Identifiers appropriate for its operation.

Different betting configurations can be created for different sets of brands and skins.

The same game can be offered in different configurations through different brands and skins. For example, the minimum bet might be higher for one brand than for another.

3.8 Channel Type & Presentation Type

The *channel type* identifies the distribution channel through which the game will be played. For example, desktop, tablet, or mobile.

The *presentation type* identifies the format in which the game will be rendered. For example, HTML5, FLASH10, or an executable.

Together, the channel type and the presentation type identify the operating environment for the game.

The RGS is expected to pick a betting configuration and deliver content that is appropriate to that operating environment. If the game is not appropriate for the specified operating environment, it should not be launched.

3.9 Locale Code & Currency Code

The *locale code* identifies the language in which the game should be presented, as well as the country variant. Locale codes are defined in ISO-639; country codes are defined in ISO-3166-1 Alpha-2. For example, English US or English Great Britain.

Typically, the locale code in the game launch URL is set to the value being used by the iGaming console. If the specified locale is not supported, a default locale is used by the game. Many games allow the player to select a different locale after the game has been launched.

The *currency code* identifies the currency in which the game will be played. Currency codes are defined in ISO-4217. For example, US Dollar, Euro, or Swedish Kroner. This is typically the currency of the player account. All monetary transactions take place in this currency.

3.10 Game Identifier

The *game identifier* uniquely identifies a specific game on an RGS; more accurately, a specific instance, or build, of a game theme.

An RGS might contain multiple instances of a game theme — a new version and an older version, or an HTML version and a Flash version. The new version might have a new set of configuration options, or the new version might be available to a different set of channel and presentation types.

The supplier of the game is responsible for assigning a globally unique game identifier to each instance.

Other properties of the game further describe the instance, such as theme, game type, package, version, release number, presentation type, channel type, etc.

In addition to these static properties of a game, the game may have many configurable options. These configurable options are used to create betting configurations and free-spin configurations for the game.

Each configuration is associated with a specific brand and skin. The RGS is responsible for selecting the appropriate betting configuration for the player when a game is launched. The RGS uses the content of the game launch URL to make that selection.

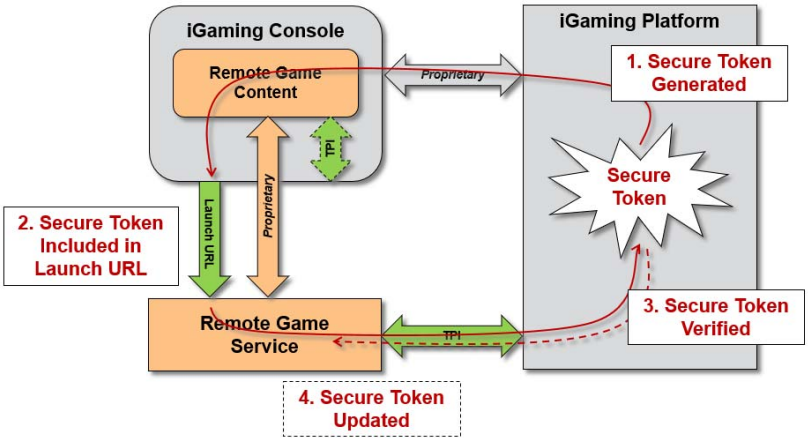
3.11 Secure Token

The *secure token* acts as an authorization key. It uniquely identifies the game session launched by the iGP. Typically, UUIDs - *Universally Unique Identifiers* - are used as secure tokens.

The secure token is included in all requests sent by the RGS to the iGP. If the iGP doesn't recognize the secure token, or the wrong game or player is associated with the secure token, the iGP should not process the request.

For added security, the iGP can change the secure token when it responds to the RGS. Because the secure token may change during a game session, a separate game session identifier is assigned by the RGS when the game is launched. The game session identifier remains constant during the game session.

The image below illustrates how the secure token is intended to be used.



Prior to launching a game, the iGP generates a secure token (#1), recording the secure token and the associated launch parameters in its database.

Next, the secure token is passed to the RGS (#2). It is passed to the RGS in the launch URL along with other launch parameters for the game.

Upon receipt of the launch URL, the RGS passes the secure token and launch parameters to the iGP for verification (#3). If the secure token and launch parameters are correct, the RGS will proceed, starting a new game session and launching the game.

In the response to the verification request, or any subsequent response, the iGP can change the secure token (#4). The RGS is expected to use the new secure token in any subsequent requests.

CHAPTER

4

Game Configuration

When a game is launched, there must be at least one betting configuration that matches the launch parameters provided by the iGC.

Various protocol-related properties are specified when the betting configuration is created. These properties are defined in the protocol. Some, such as brand and skin, appear in the game launch URL and, thus, have to do with the selection of a betting configuration. Others, such as paytable and base currency, have to do with the behavior of the game after it is launched.

Game-specific properties can also be specified. These properties are defined by the supplier of the game — they are unique to the specific game. For example, game speed or minimum bet. This information is also communicated through the protocol, but is not part of the protocol.

4.1 Game List

Game configuration starts with the game list. The game list provides information about the set of games available on the RGS. It includes the:

- static properties for the games (properties that cannot be changed)
- protocol-required properties (properties that must be configured when a betting configuration or free-spin configuration is created)
- game-specific properties (additional properties defined by the supplier).

The iGP uses the information in the game list to create betting configurations.

4.1.1 Game List Static Properties

The static properties are described in the following table. These properties are for informational purposes only. They cannot by changed by the iGP.

Table 4.1 Game List Static Properties

Property	Description
`gameId`	Globally unique identifier for the game.
`gameTitle`	Game title.
`gameDesc`	Game description.
`gameType`	Game type; selected from a standardized list.
`mfgCode`	Globally unique identifier for the manufacturer.
`themeId`	Globally unique identifier for the game theme.
`packageId`	Software package used to build the game.
`packageVersion`	Software package version.
`releaseNum`	Release number assigned to the build.

Table 4.1 Game List Static Properties

Property	Description
`channelArray`	Distribution channels and presentation types.
`localeArray`	The locales (languages) supported by the game.
`jurisdictionArray`	Jurisdictions where the game is approved.
`controllerLinkArray`	Jackpots to which the game is linked.

4.1.2 Game List Protocol-Required Properties

The protocol-required properties are described in the following table. The iGP can select from these properties when a betting configuration is created.

Table 4.2 Game List Protocol-Required Properties

Property	Description
`paytableArray`	The paytables available for the game.
`skinArray`	Brands and skins to which the game is restricted.
`currencyArray`	Currencies to which the game is restricted.

A paytable, brand, skin, and currency must be specified when a betting configuration is created. If no brands, skins, or currencies are specified in the game list, there are no restrictions when creating the betting configuration — any brand, skin, or currency can be used.

Even though a currency is specified when a betting configuration is created, the game can be played in any of the currencies contained in the `currencyArray`. If the `currencyArray` is empty, the game can be played in any currency.

The currency specified in the betting configuration is the base currency for the betting configuration. It is the currency in which monetary values in the betting configuration are denominated. If the game is played in another currency, currency multipliers are used to adjust the betting configuration to that currency.

4.1.3 Game List Game-Specific Properties

The game list also contains two sets of game-specific properties: game-specific configuration properties and game-specific launch parameters. They are described in the following table.

Table 4.3 Game List Game-Specific Properties

Property	Description
Configuration Properties	
`configSchema`	URL of the JSON schema that defines the properties.
`configHash`	SHA-1 hash value for the JSON schema.
Launch Parameters	
`parameterArray`	Additional parameters for the Game Launch URL.

Game-specific configuration properties must be included when a betting configuration is created.

Game-specific launch parameters must be appended to the game launch URL when the game is launched.

The `configSchema` property contains the URL of the JSON schema that defines the game-specific configuration properties.

The `configHash` property contains the SHA-1 hash value for the JSON schema. The iGP can use this value to verify that the correct schema was loaded.

4.2 Betting Configurations

Creating a betting configuration is easy. The iGP simply specifies the protocol-required configuration properties and the game-specific configuration properties for the game. These properties and brief descriptions are displayed in the table below.

Table 4.4 Betting Configuration Properties

Property	Description
`gameId`	Globally unique identifier for the game.
`betConfigTitle`	Betting configuration title.
`betConfigDesc`	Betting configuration description.
`brandId`	Brand identifier.
`skinId`	Skin identifier.
`segmentId`	Player Segment identifier.
`paytableId`	Paytable identifier.
`baseCurrency`	Base currency code.
`configData`	Game-specific configuration object.

4.2.1 Player Segments

Player segments can be used by the iGP to divide players into different marketing groups within the same brand and skin. Even though the brand and skin are the same, different betting configurations can be offered to different groups of players. For example, low-denomination betting configurations can be offered to some players; and, high-denomination betting configurations can be offered to other players.

4.2.2 Betting Configuration States

To avoid confusion, there can only be one betting configuration for a specific combination of brand, skin, and player segment.

If a new betting configuration is a duplicate of an existing betting configuration, the new betting configuration replaces the old one. Old betting configurations are automatically archived.

The image below illustrates all possible betting configuration states.

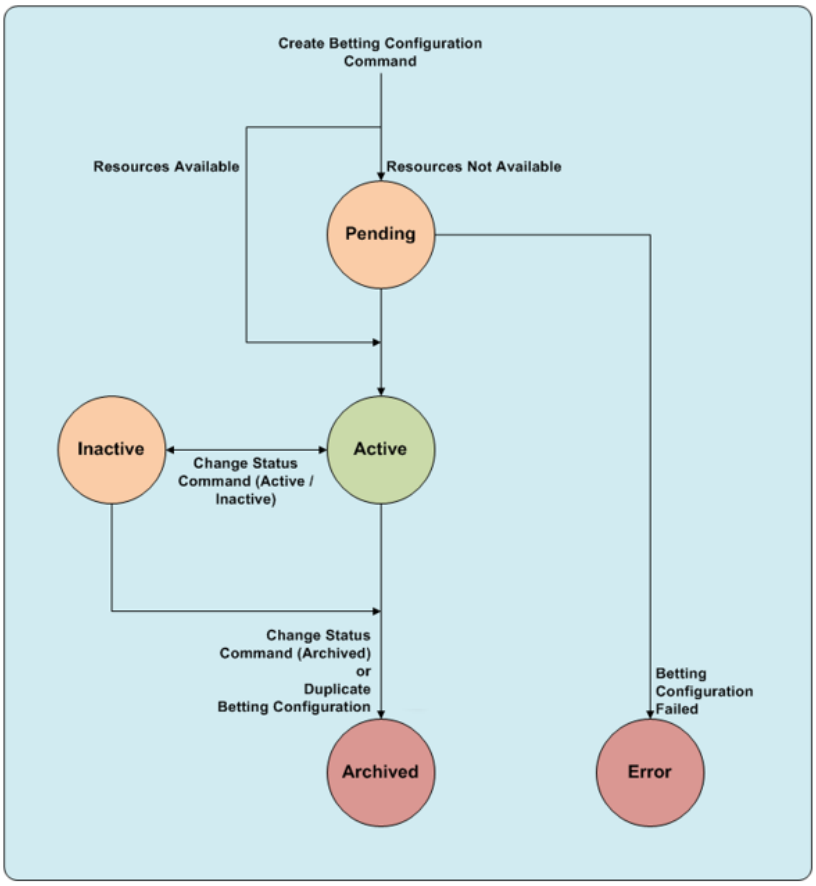
Initially, when the betting configuration is first created, it enters either the *pending* or *active* state. If all resources necessary to launch the game are available, the betting configuration goes directly to the active state.

If some of the resources are not available – for example, they may need to be retrieved from a game repository – the betting configuration will enter the pending state. Once the resources are available, the betting configuration will transition to the active state. If the resources cannot be retrieved, the betting configuration will enter the *error* state.

When the new betting configuration becomes active, if an existing betting configuration for the same brand, skin, and player segment exists, the existing betting configuration will automatically be archived — it will transition to the *archived* state.

Once a betting configuration is in the active state, the iGP can move the betting configuration between the *active* and *inactive* states. The iGP can also manually archive the betting configuration.

Betting configurations will only be offered to players if they are in the active state.



Each betting configurations is assigned a unique identifier. It is reported in the response to the request to create the betting configuration.

In its response, the RGS also reports the:

- Base URL of the game, the launch parameters are appended to this URL
- Help URL, the location of the games help files
- Demo URL, the location of the ‘demo’ version of the game
- History URL, the location of the file containing the history of betting configurations for the game.

4.2.3 Currency Multipliers

The base currency is the currency in which monetary values in a betting configuration are specified. In some cases, the monetary values of a particular currency might be appropriate for other currencies. For example, if monetary values are specified in Euros,

those values might also be appropriate when the game is played in Dollars or Pounds. The currencies are of a similar magnitude.

In other cases, the currencies may not be of a similar magnitude. Currency multipliers are used when a game is configured in one currency but played in a currency of a different magnitude.

For example, if the betting configuration is in Euros and the currency multiplier from Euros to Kroner is 10, when the game is played in Kroner, all monetary values in the betting configuration are multiplied by 10. For example, a minimum bet of 25 Euros would become 250 Kroner.

Currency multipliers are configured by brand and skin. If no currency multiplier is defined for a brand, skin, and currency, then a currency multiplier of 1 is used.

4.3 Free-Spin Configurations

Certain account balances may only be available as free spins. The player sees the number of free spins (or free bets) rather than the monetary value.

In the background, there is a monetary value associated with each free spin — the wager value of the free spin.

Free spins are associated with free-spin configurations. Just as games have betting configurations, they also have free-spin configurations.

4.3.1 Free-Spin Configuration

Free-spin configurations are similar to betting configurations. The RGS assigns a free-spin configuration identifier to each free-spin configuration requested by the iGP.

The free-spin configuration identifier is contained in the response to the request to create the free-spin configuration.

Before free spins can be used, the game must be configured for free spins. At least one free-spin configuration must be created. The configuration used for free spins may be different than the configuration used for standard game play.

As with betting configurations, some protocol-required properties must be specified when creating a free-spin configuration. Additionally, game-specific properties can be specified. These properties are defined by the supplier of the game.

There are no game-specific launch parameters for free spins.

4.3.2 Creating Free-Spin Configurations

The RGS assigns a free-spin configuration identifier to each free-spin configuration. The RGS also reports the `baseFreeSpinValue` (the wager value of one free spin).

The RGS should offer free spins to the player if the game, brand, and skin of the free-spin configuration match the betting configuration being played.

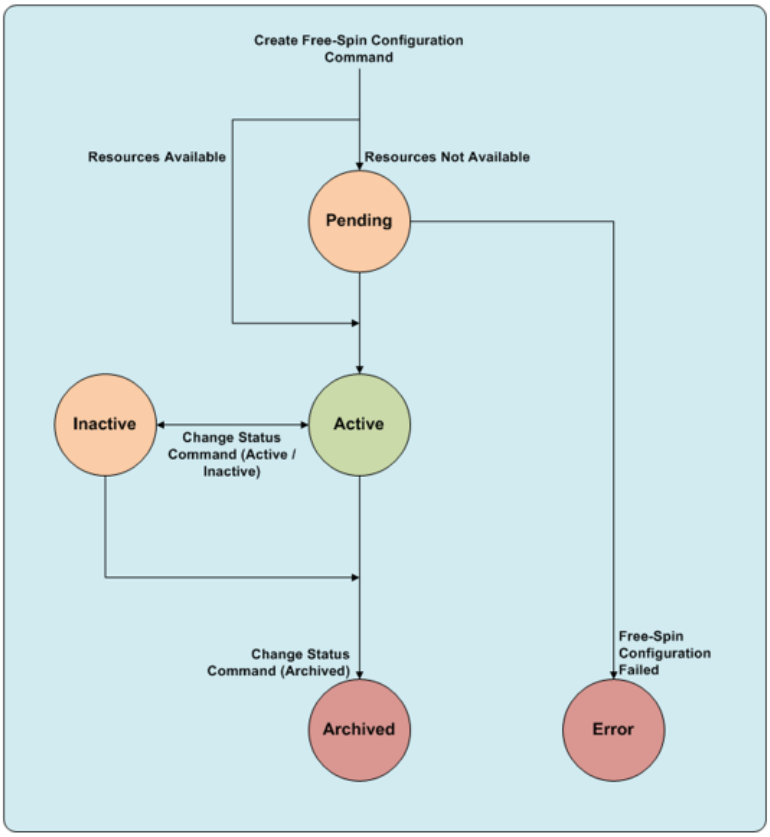
The properties specified for a free-spin configuration are listed in the table below.

Table 4.5 Free-Spin Configuration Properties

Property	Description
`gameId`	Globally unique identifier for the game.
`freeSpinTitle`	Free-spin configuration title.
`freeSpinDesc`	Free-spin configuration description.
`brandId`	Brand identifier.
`skinId`	Skin identifier.
`paytableId`	Paytable identifier.
`baseCurrency`	Currency code.
`freeSpinData`	Game-specific configuration object.

There can be multiple free-spin configurations for a game with a specific combination of brand and skin. There are no default free-spin configurations. The iGP must always explicitly identify the free-spin configuration that the RGS should use. Also, the iGP must explicitly archive obsolete free-spin configurations.

The diagram below illustrates the states a free-spin configuration might enter.



4.3.3 Reporting Free Spins

When a free spin is used, the RGS treats the free spin as if it were a standard monetary transaction, reporting the amount wagered and the amount won.

The RGS also includes the free-spin configuration identifier in the monetary transactions. This tells the iGP that free spins were used.

Free spins and conventional wagers can both be used at the same time. Typically, this won't happen with spinning reel games, but it may happen with table games. *Free spins* are often called *free chips* when used with table games; in some games, *free chips* and *cash chips* can both be bet on the same hand.

CHAPTER

5

Game Operations

5.1 Game Play Operations

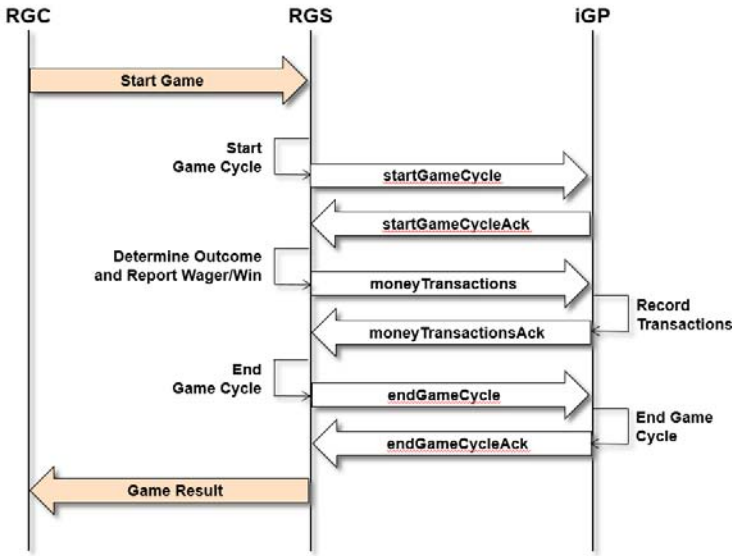
5.1.1 Simple Game Play

The sequence diagram below illustrates how commands in TPI are used to report simple game play – for example, a simple spinning reel game.

The sequence starts with the Remote Game Content (RGC) making a start-game request to the RGS. Based on this request, the RGS starts a game cycle and then reports it to the iGP using a `startGameCycle` command. The iGP responds with a `startGameCycleAck`.

Next, the RGS determines the game outcome and reports the wager and win to the iGP in a `moneyTransactions` command. The iGP responds with the `moneyTransactionsAck` command. The RGS then ends the game cycle, notifying the iGP and sending the game results to the RGC for display to the player.

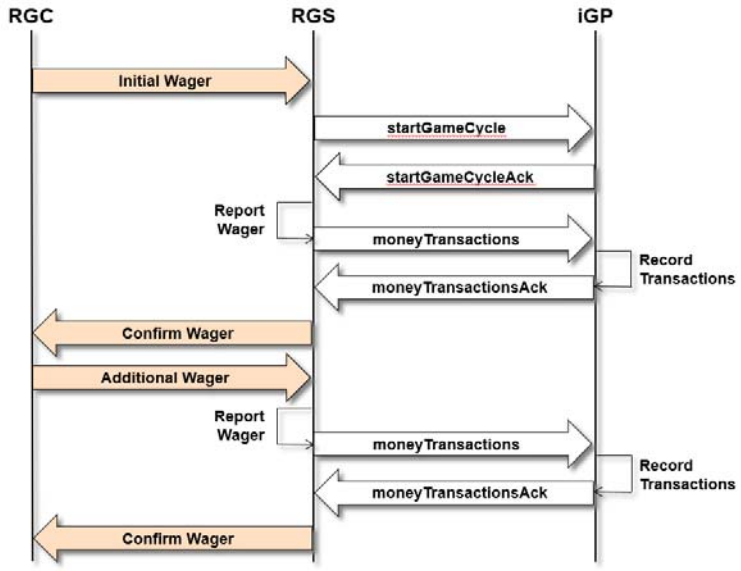
The transaction day is assigned when the monetary transactions are recorded by the iGP; the revenue day is assigned when the game cycle ends.



5.1.2 Complex Game Play

The sequence diagram below illustrates a more complex game play scenario where the wagers and wins are spread across multiple monetary transactions and, possibly, multiple accounting days. Like the previous example, the sequence starts with the RGC making a start-game request to the RGS.

In response, the RGS starts a game cycle; notifies the iGP that the game cycle has started; sends the initial wager to the iGP for approval; and then confirms the wager to the RGC.

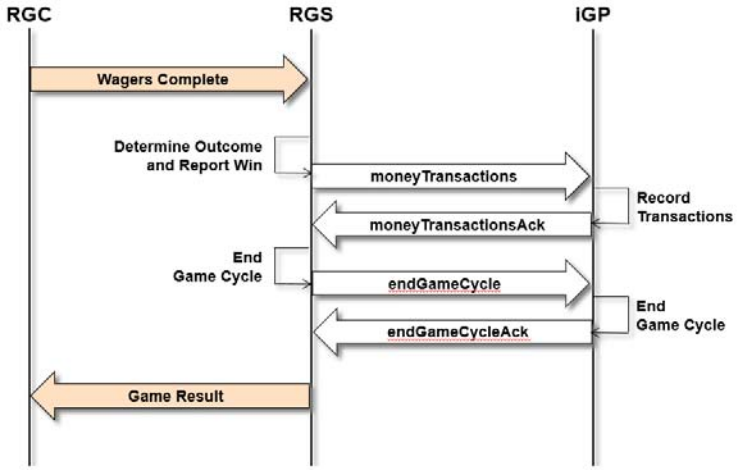


Next, the RGC reports another wager by the player. The RGS reports the wager to the iGP and then confirms the wager back to the RGC. This pattern could continue for many more wagers.

Finally, the RGC indicates that the wagers are complete.

The RGS determines the outcomes and reports the win to the iGP. The RGS then ends the game cycle; notifies the iGP that the game cycle has ended; and, then sends the game results to the RGC for display to the player.

This example is very much like the simple game play example, except the wagers and wins are spread across multiple monetary transactions. In fact, they could have been spread across multiple accounting days. The wagers could have been made on one accounting day and the wins could have been determined on another accounting day. This is often the case with sports betting.



5.2 Reconciliation Commands

Reconciliation commands are used to complete interrupted game cycles. They are the same as other commands, except no secure token is included.

A reconciliation command assumes that the game session has ended and that the secure token has been discarded, so the secure token is omitted. A game session and secure token are required to start a game cycle, but not to complete and end the game cycle.

5.3 Unfinished Game Cycles

Given the nature of online gaming, unfinished game cycles are inevitable! The player closes the browser. The internet connection is lost. Or, in the case of sports betting, the game outcome won't be determined until some point in the future.

In TPI, the RGS has several choices when confronted with an unfinished game cycle:

- First, void the game cycle and cancel any associated monetary transactions.
- Second, auto-complete the game cycle and post any additional monetary transactions required to finish the game cycle. This is what happens with sports betting.
- Third, suspend the game cycle and let the player complete it later. This is what might happen with a blackjack game.

If a game cycle is suspended, it remains open until:

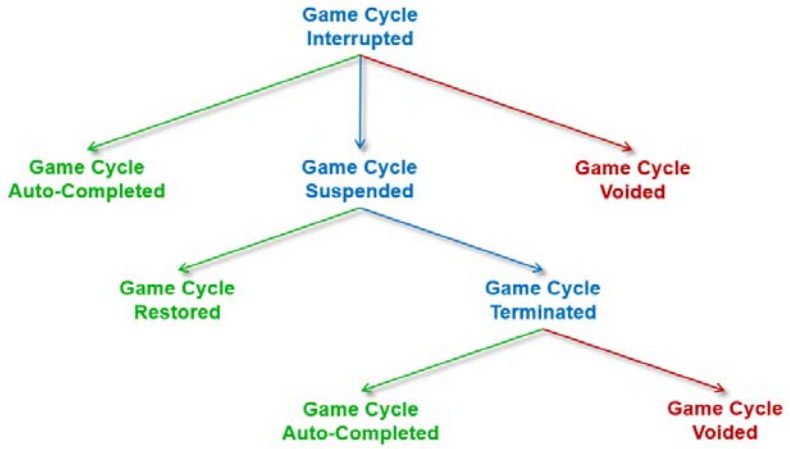
- the player returns and completes the game cycle,
- the RGS terminates the game cycle. or
- the iGP directs the RGS to terminate the game cycle.

Once a game cycle is terminated, the game cycle must be auto-completed or voided.

Game design, as well as operational and jurisdictional requirements, will determine whether interrupted game cycles should be voided, auto-completed, or suspended.

Likewise, game design, as well as operational and jurisdictional requirements, will determine whether terminated game cycles should be voided or auto-completed.

The illustration below shows the options available to the RGS.



After a game cycle is interrupted, three initial options are available to the RGS: one, auto-complete the game cycle – finish the game cycle for the player using an optimal strategy; two, suspend the game cycle – let the player finish it later; or, three, void the game cycle – cancel all wagers and wins.

A suspended game cycle has two additional options: The game cycle can be restored so that the player can finish it; or, the game cycle can be terminated – this decision could be made by the RGS or the iGP. Typically, it is made after some period of time has passed – for example, 30 days.

If the decision is to terminate the game cycle, two more options become available: auto-complete the game cycle or void the game cycle.

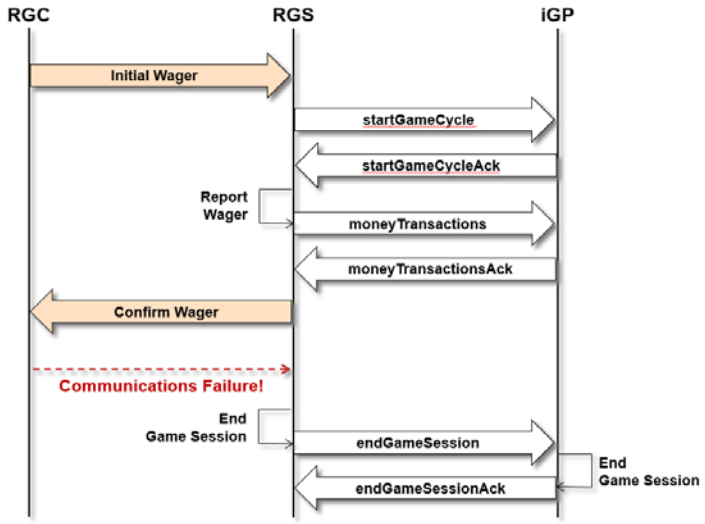
As mentioned before, game design, as well as operational and jurisdictional requirements, will determine which path is followed.

5.4 Interrupted Games

The sequence diagram below illustrates how interrupted game cycles are expected to be handled in TPI — in this case, when the game is auto-completed. The sequence would be very similar if the game cycle was voided.

In this example, a game cycle has been started and a wager has been recorded. Then, there is a communications failure — the game content has lost communications with the RGS and the RGS has decided to end the game session.

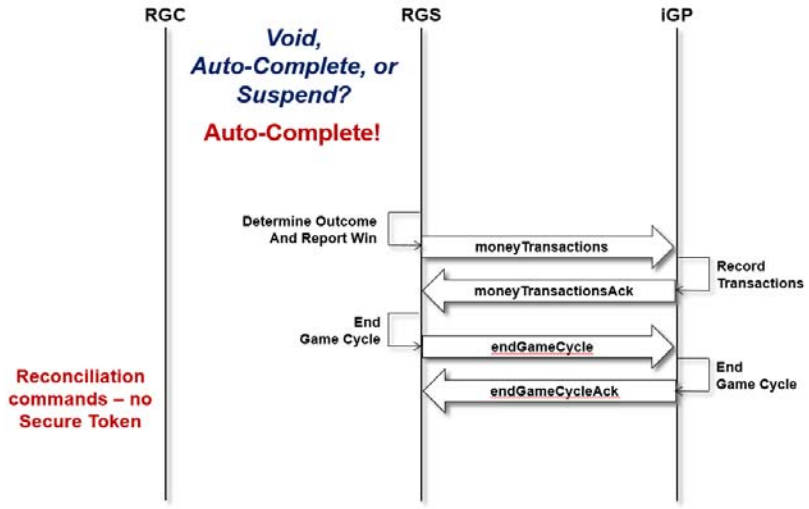
This may not happen immediately — the RGC and the RGS may try to reestablish communications before the RGS ends the game session. However, if the condition persists, eventually, the RGS will have to give up and end the game session.



Once the game session has ended, the RGS needs to make a decision — void, auto-complete, or suspend the interrupted game cycle. In this case, the decision is to auto-complete the game cycle.

The following sequence diagram illustrates this process.The RGS determines the game outcome, usually following an optimal game strategy, reports any winnings to the iGP, and then ends the game cycle. The `moneyTransactions` and `endGameCycle` commands are sent as reconciliation commands — the secure token is omitted. An exception code is included to explain what happened.

A similar sequence would be followed if the decision was to void the game cycle. In that case, a `cancelTransactions` command would be used instead of the `moneyTransactions` command.


RGC
RGS
iGP
Void,
Auto-Complete, or
Suspend?
Auto-Complete!
Determine Outcome
And Report Win
moneyTransactions
Record
Transactions
moneyTransactionsAck
End
Game Cycle
endGameCycle
End
Game Cycle
endGameCycleAck
Reconciliation
commands – no
Secure Token

5.5 Restoring Game Cycles

Restoring suspended game cycles is a little more complicated.

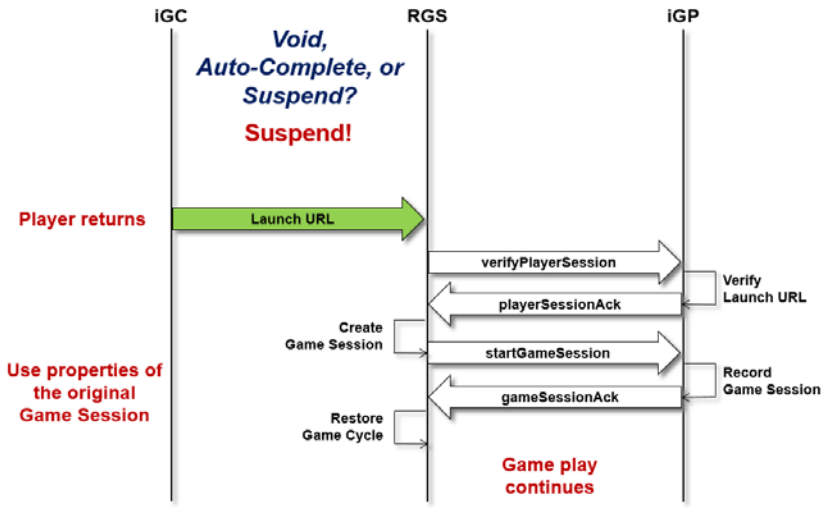
The player must be able to finish the restored game cycle under the same operating conditions as were in place when the game cycle was started — this is important for player protection as well as revenue reporting.

However, some of the parameters of the new game launch URL may be different than the parameters of the original game session. Thus, the RGS may have to substitute some parameters of the launch URL with parameters of the original game session. That complicates the process.

A new game session must be started when the player returns to the game. However, many of the properties of the original game session must be used for the new game session. These include the game, brand, skin, betting configuration, player, player account, currency code, affiliate, and jurisdiction.

Other properties of the new game session may be different, such as the distribution channel or presentation format.

This example demonstrates what should happen when a player returns to finish a suspended game cycle.



First, the iGC should send the game launch URL to the RGS as usual. The RGS should then verify the game launch URL as usual — the set of launch parameters contained in the launch URL should be sent to the iGP for verification.

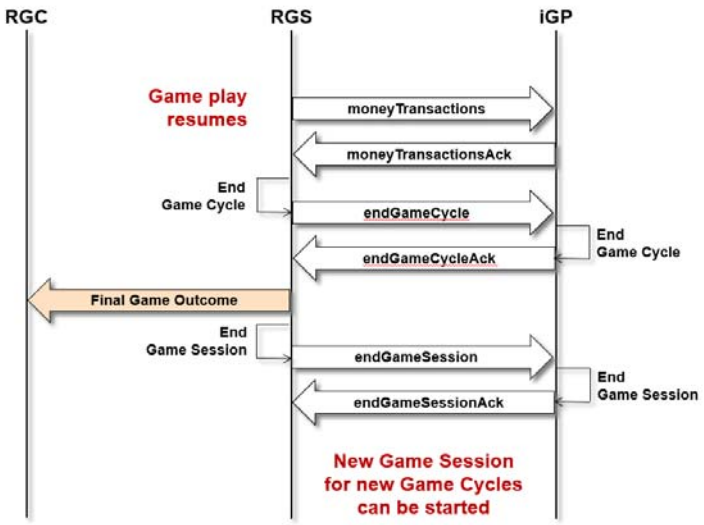
Next, the RGS should start a new game session. However, instead of using the current launch parameters and the current player session information, the RGS must use the parameters of the original game session. Then, the game cycle can be restored, and game play can continue.

After the player finishes the suspended game cycle, the game session must be ended. Game play should not continue using the restored game session.

If the player wants to continue playing the game, a new game session should be started using the parameters from the current game launch URL and the current player session. The game launch URL should be reverified and then the new game session should be started.

The following sequence diagram illustrates the steps that should be followed to finish the suspended game cycle.

First, any monetary transactions required to finish the game cycle should be reported to the iGP. Next, the game cycle should be ended, the iGP should be notified, and the final results of the game should be sent to the RGC. Lastly, the game session should be ended. If the player wants to continue playing, a new game session for the new game cycles should be started.



5.6 Terminating Game Cycles

If necessary, the iGP can force the RGS to terminate a suspended game cycle. The RGS might also have operational or jurisdictional rules in place for terminating suspended game cycles – for example, after 30 days, suspended game cycles must be terminated.

If the iGP chooses to terminate a game cycle, it has two options:

The iGP can instruct the RGS to finish the game cycle — the game cycle should be voided or auto-completed as appropriate. Game design, as well as operational and jurisdictional requirements, will determine whether the terminated game cycle should be voided or auto-completed.

Or, the iGP can instruct the RGS to cancel the game cycle — the game cycle should be voided. All monetary transactions are cancelled.

5.7 Ending a Game Session

Game sessions typically end when the player leaves the game – voluntarily by exiting the game, or involuntarily because of a browser failure, network interruption, etc. Game sessions may also end for operational or jurisdictional reasons.

For example, the operation may prefer to start a new game session at the beginning of each accounting day; or, the jurisdiction may require that the RGS end a game session if there is no activity for 30 minutes.

When a game session ends, the RGS needs to notify the iGP that the game session has ended. Once a game session has ended, no additional game cycles can be started.

5.8 Closing a Game Session

The iGP can force the RGS to close a game session. For example, the jurisdiction may require that the game session be closed if the player leaves the jurisdiction; or, the jurisdiction may require that the game session be closed if the player exceeds a responsible gaming limit.

In such cases, the iGP has four options:

- Interrupt the current game cycle, end the game session, and then;
  - start a new game session, or
  - do not start a new game session
- Finish the current the game cycle, end the game session, and then;
  - start a new game session, or
  - do not start a new game session

If the decision is to interrupt the game cycle, the game cycle is immediatly auto-completed, voided, or suspended as appropriate for the game, operation, and jurisdiction.

If the decision is to finish the current game cycle, the player is allowed to finish the game cycle beofre the game session is closed.

The iGP simply identifies the player, player account, and game session, and then indicates whether the game session should be closed immediately — and the game interrupted — or whether the RGS should wait until the current game is finished.

The iGP also indicates whether a new game session should be opened after the initial action is taken.

CHAPTER

6

Jackpot Configuration

6.1 Introduction

Jackpot controllers and jackpot levels are configured on the PJS. The RGS and iGP can read the jackpot configuration information but cannot change it. TPI does not contain commands for configuring jackpots, only for reporting the configurations.

The links between games and jackpot levels are configured on the RGS. Like the jackpot configuration information, the PJS and iGP can read the jackpot link information, but cannot change it. TPI does not contain commands for configuring jackpot links, only for reporting the links. Jackpot link information is available from the PJS or the RGS. When requested from the RGS, the information is contained in the game list with other information about the game.

The image below illustrates the relationships between progressive jackpot services, jackpot controllers, and jackpot levels.

In this example, the PJS contains three controllers: jackpot controllers 1, 2, and 3. Jackpot controller 1 is a standard progressive controller; jackpot controllers 2 and 3 are mystery controllers. Jackpot controller 1 has five jackpot levels; controller 2 has two levels; and controller 3 has one level.

There are no restrictions on how many controllers can be supported by a PJS or how many levels are within a controller.


Progressive Jackpot Service
Jackpot Controller #1
Type = Standard
Jackpot Level #1
Jackpot Level #2
Jackpot Level #3
Jackpot Level #4
Jackpot Level #5
Jackpot Controller #2
Type = Mystery
Jackpot Level #1
Jackpot Level #2
Jackpot Controller #3
Type = Mystery
Jackpot Level #1

6.2 Standard Jackpot Configuration

The table below displays jackpot configuration properties for standard jackpots.

Table 6.1 Standard Jackpot Configuration Properties

Property	Description
`levelId`	Jackpot level identifier.
`levelTitle`	Jackpot level title.
`levelOdds`	Intended odds of hitting the jackpot level.
`levelWagerAmt`	Intended minimum wager to hit the jackpot level.
`resetValue`	Reset value after a hit.
`incrementMin`	minimum percentage of wagers contributed.
`incrementMax`	maximum percentage of wagers contributed.
`maxValue`	maximum value of the jackpot level.

6.3 Mystery Jackpot Configuration

The table below displays jackpot configuration properties for mystery jackpots.

Table 6.2 Mystery Jackpot Configuration Properties

Property	Description
`levelId`	Jackpot level identifier.
`levelTitle`	Jackpot level title.
`resetValue`	Reset value after a hit.
`incrementMin`	Minimum percentage of wagers contributed.
`incrementMax`	Maximum percentage of wagers contributed.
`maxValue`	Maximum value of the jackpot level.
`mysteryMin`	Minimum for the randomly selected value.
`mysteryMax`	Maximum for the randomly selected value.

6.4 Jackpot Links

The same game may be offered to players through many different brands and skins. The same game may also be offered with different paytables. These different game configurations can be targeted towards different sets of players; they might also be associated with different operators. Thus, it might not make sense to offer the same set of jackpots to all players of a game.

TPI allows different groups of jackpots to be associated with different sets of brands, skins, and paytables. One group of jackpots can be associated with one set of brands, skins, and paytables while another group of jackpots can be associated with another set.

In TPI, a group of jackpots is a jackpot controller. Thus, in TPI, jackpot controllers are associated with specific sets of brands, skins, and paytables. In this way, different groups of jackpots can be offered to different groups of players, even though they are all playing the same game.

The player is eligible to win the jackpots associated with the brand, skin, and paytable of the current betting configuration.

Typically, contributions follow the links. If a player is playing a game that is linked to a jackpot, typically, the game contributes to the jackpot. The game needs to know the links so that it can contribute correctly.

6.4.1 Standard Jackpot Link

For standard jackpots, jackpot levels are linked to winning combinations within the game. The properties of the jackpot links include:

- the jackpot level identifier
- the actual minimum wager required to hit the jackpot level
- the actual odds of hitting the jackpot level
- the standard paytable win if the winning combination was not linked to the jackpot

Note that the winning combination itself is not reported through the protocol. Only the minimum wager, odds, and standard paytable win are reported — the information that is important for verifying that the jackpot links were properly configured.

6.4.2 Mystery Jackpot Links

For mystery jackpots, jackpot levels are mapped to specific wager denominations. The links simply identify the jackpot level and the denomination — the wager denomination eligible to win the jackpot level. The value `-1` indicates that all denominations are eligible to win the jackpot level.

6.5 Currency Exchange

One of the complicating factors for jackpots is that the currency in which a game is being played may be different than the currency in which a jackpot controller is denominated. Because player accounts can be denominated in different currencies, games can be launched in many different currencies. However, jackpots only have a single currency. Thus, currency exchange is needed.

In TPI, currency exchange is handled by the PJS.

Incoming data from the RGS is always in the currency in which the game is being played. Outgoing data from the PJS is reported in both the currency in which the game is being played and the base currency of the jackpot controller, if it is different.

CHAPTER

7

Jackpot Operations

7.1 Qualifying Wagers

When a game is played, the RGS reports the qualifying wagers from the game to the PJS. Typically, wagers on games that are linked to a jackpot contribute to that jackpot. However, other methods may be used. The RGS is responsible for picking the method and determining which wagers to report as qualifying wagers.

Qualifying wagers are reported by jackpot controller, not jackpot level. The PJS is responsible for calculating the contribution to each jackpot level within the jackpot controller and incrementing the jackpot values.

If a game is linked to multiple jackpot controllers, the same wager may be a qualifying wager for more than one jackpot controller.

If a game cycle is voided, the qualifying wagers should also be voided. The RGS is responsible for notifying the PJS when qualifying wagers should be voided.

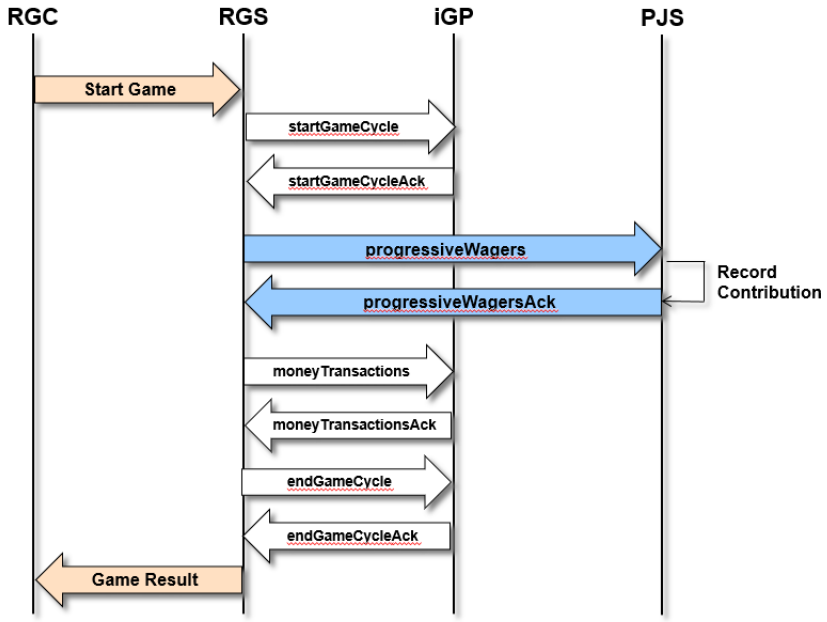
The sequence diagram below illustrates how qualifying wagers from a game cycle are intended to be reported. This example shows the qualifying wagers being reported inside the game cycle - while the game cycle is still open. The protocol does not require this behavior. The qualifying wagers can also be reported outside the game cycle — after the game cycle has ended. The choice is left to the RGS.

Reporting qualifying wagers inside the game cycle ties the contributions closer to the game cycle. The RGS can assure that the contributions were made before allowing the game cycle to continue. It may also make the awarding of mystery jackpots easier.

Mystery jackpot hits are reported in the acknowledgment to the qualifying wagers. If the game cycle is still open, the mystery jackpot can be paid immediately as part of the open game cycle. If the game cycle is not open, the RGS may have to wait for a new game cycle to start before it can pay the mystery jackpot.

On the other hand, reporting qualifying wagers outside the game cycle can improve performance — the game cycle is not blocked while the qualifying wagers are being reported.

The choice of methodologies is left to the RGS supplier to decide.



The properties used to report qualifying wagers include: the transaction identifier - this is assigned by the RGS; the jackpot controller identifier; and, the wager amount - the value of the qualifying wagers.

When qualifying wagers for mystery jackpots are reported, two additional properties are included. These properties are intended to help the PJS select mystery jackpot winners. The two properties are the denomination wagered for the game cycle and the base paytable win for the game cycle.

For example, a minimum win might be required to win the mystery jackpot. Or, the mystery jackpot might not be awarded if the base paytable win plus the jackpot value exceed a jurisdictional limit. In many cases, the two additional properties are not used. They are included in the protocol in case they are needed.

7.2 Jackpot Contributions

The response from the PJS contains the amount contributed to each jackpot controller. The properties are displayed in the table below.

Table 7.1 Jackpot Contribution Properties 1

Property	Description
`contribAmt`	The amount contributed to the jackpot controller.
`referenceId`	Reference identifier; assigned by the PJS.
`transDay`	Transaction day; accounting day of the PJS.
`contribExc`	Exception code; only included if no contributions were made to the jackpot controller.

If the game is being played in one currency and the jackpot controller is denominated in a different currency, the value of the contribution is also specified in the currency of the jackpot controller. In such cases, the PJS also includes the currency code in which the jackpot controller is denominated; the exchange rate from the currency of the jackpot controller to the currency of the game; and the value of the contribution in the currency of the jackpot controller.

Table 7.2 Jackpot Contribution Properties 2

Property	Description
`foreignCode`	Currency code in which the jackpot controller is denominated.
`foreignRate`	The exchange rate from the currency of the jackpot controller to the currency of the game.
`foreignAmt`	The value of the contribution in the currency of the jackpot controller.

7.3 Jackpot Values

The response from the PJS containing the jackpot contributions also includes the current values of the jackpots.

Benefit:
These five properties are the same properties used in other GSA standards for land-based jackpot.

Table 7.3 Jackpot Value Properties 1

Property	Description
`controllerId`	Jackpot controller identifier.
`levelId`	Jackpot level identifier.
`currentSeq`	A strictly increasing sequence used to sequence updates to the value of the jackpot level.
`currentAmt`	Current value of the jackpot level.
`currentText`	Prize description or text representation of the current value of the jackpot level.

These five properties are the same properties used in other GSA standards for land-based jackpots.

If the game is being played in one currency and the jackpot controller is denominated in a different currency, the jackpot values are also specified in the currency of the jackpot controller. In such cases, three additional properties are included: `foreignCode`, `foreignRate`, and `foreignAmt`.

Table 7.4 Jackpot Value Properties 2

Property	Description
`foreignCode`	Currency code in which the jackpot controller is denominated.
`foreignRate`	The exchange rate from the currency of the jackpot controller to the currency of the game.
`foreignAmt`	The value of the jackpot in the currency of the jackpot controller.

7.3.1 Monetary Transactions

Along with wagers and wins, jackpot contributions can be relayed by the RGS to the iGP through monetary transactions.

When this is done, additional information about the jackpot contributions is included in the monetary transactions: the progressive jackpot service

identifier; the jackpot controller identifier; and, the transaction day - the accounting day of the jackpot controller.

This is a case where the *special* transaction type is used. Special transactions do not affect the account balance of the player. The contributions are reported as part of the monetary transactions for the game, but do not affect the player's account balance.

7.4 Standard Jackpot Wins

The process for awarding standard jackpot wins takes place in four steps. It starts with a winning combination being hit on an RGS.

First, the RGS notifies the PJS that the jackpot has been hit. This command includes the last jackpot value received by the RGS.

Second, the PJS responds with the amount that should be paid. This may be different than the last value received by the RGS — there might have been additional contributions or the jackpot value might be rounded up to the next major unit of the currency.

Third, the RGS makes the payment and then notifies the PJS that the payment has been made. The amount paid might be slightly different than the amount authorized by the PJS — the RGS might have rounded the amount to an even multiple of the wager.

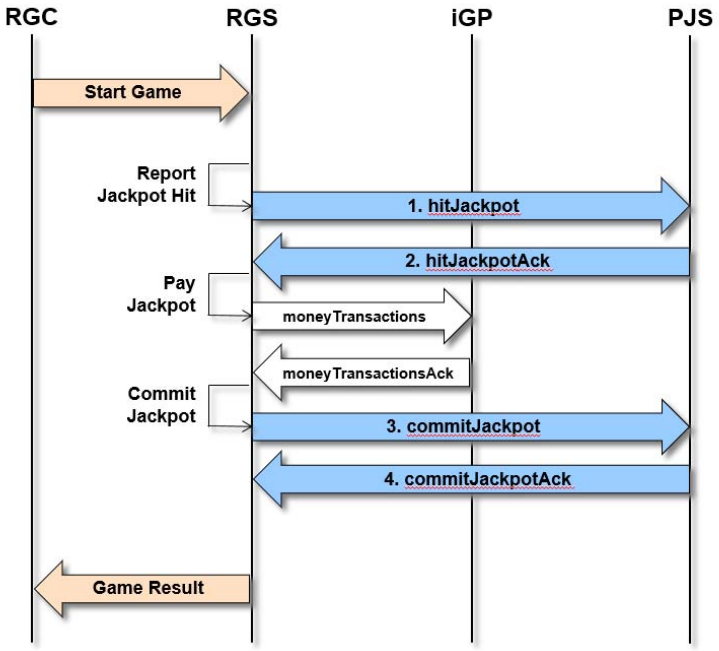
Fourth, the PJS acknowledges the payment. The process is complete.

This sequence diagram illustrates the process. The commands to start and end the game cycle have been omitted due to space constraints — they are still part of the overall process.

The first thing that the RGS does after determining that a winning combination has been hit, is to send a `hitJackpot` command to the PJS. This tells the PJS that there has been a hit. It also includes the last jackpot value received by the RGS.

The PJS responds with a `hitJackpotAck` command containing the amount that the RGS is authorized to pay the player. The RGS then uses standard monetary transactions to pay the jackpot to the player.

After the payment has been made, the RGS uses the `commitJackpot` command to report the amount actually paid to the PJS. The PJS responds with a `commitJackpotAck` command.



If an error occurred and the jackpot could not be paid, the same four-step sequence would be followed. However, rather than including the actual amount paid, the `commitJackpot` command would include an exception code indicating why the payment was not made.

7.4.1 Standard Jackpot Hit

The properties used to report a standard jackpot hit are displayed in the table below.

Table 7.5 Standard Jackpot Hit Properties

Property	Description
`controllerId`	Jackpot controller identifier.
`levelId`	Jackpot level identifier.
`transId`	Transaction identifier; assigned by the RGS.
`hitSeq`	Jackpot sequence number at the time of the hit.
`hitAmt`	Jackpot value at the time of the hit.
`hitText`	Jackpot text representation at the time of the hit.

7.4.2 Standard Jackpot Response

The response from the PJS to the jackpot hit includes the properties displayed in the table below.

Table 7.6 Standard Jackpot Response Properties 1

Property	Description
`referenceId`	Reference Identifier; assigned by the PJS.
`transDay`	Transaction Day; accounting day of the PJS.
`awardSeq`	Jackpot sequence number at the time of the award.
`awardAmt`	Jackpot value at the time of the award.
`awardText`	Jackpot text representation at the time of the award.
`isInKind`	Indicates whether the jackpot was an in-kind (non-cash) prize.

In-kind prizes typically get posted to player accounts as special transactions — the value is not added to the player account.

If the game is played in one currency and the jackpot controller is denominated in a different currency, the value of the jackpot award is also specified in the currency of the jackpot controller. Refer to the table shown below.

Table 7.7 Standard Jackpot Response Properties 2

Property	Description
`foreignCode`	Currency code in which the jackpot controller is denominated.
`foreignRate`	Exchange rate from the currency of the jackpot controller to the currency of the game.
`foreignAmt`	Value of the jackpot award in the currency of the jackpot controller.

7.4.3 Monetary Transactions

When a jackpot win is reported to the iGP through a monetary transaction, additional information about the jackpot is included.

Table 7.8 Jackpot Win Transaction Properties

Property	Description
`pjsId`	Progressive jackpot service identifier.
`controllerId`	Jackpot controller identifier.
`levelId`	Jackpot level identifier.
`pjsDay`	Transaction day; accounting day of the jackpot controller.

7.4.4 Jackpot Payment

After the jackpot win has been paid by the RGS, the RGS must report the final result to the iGP. The final result must also be reported if the payment failed, in which case a non-zero exception code must be included.

Table 7.9 Jackpot Payment Properties

Property	Description
`paidAmt`	Actual amount paid to the player.
`jackpotExc`	Exception code; only included if the jackpot was not paid.

The exception code can be omitted if the payment was successful.

7.5 Mystery Jackpot Wins

The one big difference between standard jackpots and mystery jackpots is that standard jackpot wins are determined by the RGS — the game — mystery jackpot wins are determined by the PJS — the controller. In TPI, mystery jackpot wins are announced by the PJS — the controller — in the response to the qualifying wagers command. It's a five-step process.

1. First, the PJS notifies the RGS that a jackpot should be awarded. This is in the response to the qualifying wagers.
2. Second, the RGS confirms receipt of the jackpot award.
3. Third, the PJS acknowledges that the RGS should proceed with the award.
4. Fourth, the RGS makes the payment and then notifies the PJS.
5. Fifth, the PJS acknowledges the payment.

The second, third, fourth, and fifth steps are really the same as the four steps used to award standard jackpots. In TPI, the same set of commands are used for both mystery jackpots and for standard jackpots. The mystery jackpot hit is reported in the response to the qualifying wagers. Then, the same four steps are followed to manage the jackpot award.

This sequence diagram illustrates how mystery jackpot wins are managed.

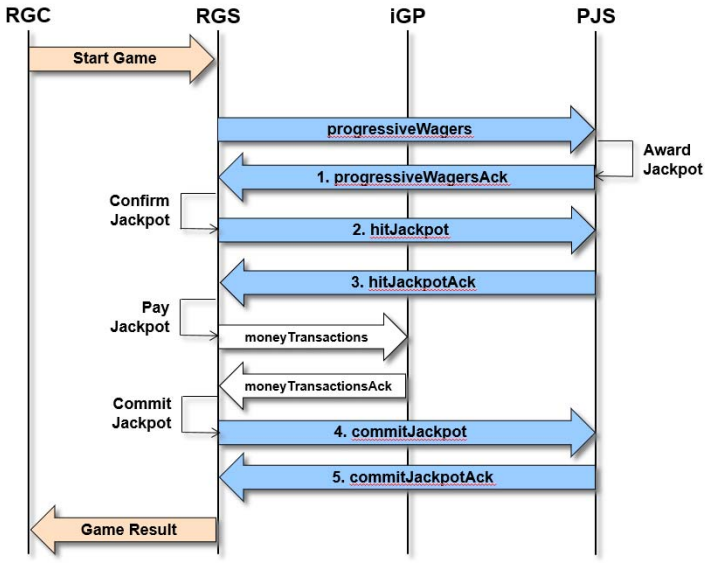
In the response to the qualifying wagers — the `progressiveWagersAck` command — the PJS indicates that the player has won a mystery jackpot.

The RGS confirms receipt of the mystery jackpot by sending a `hitJackpot` command back to the PJS. The PJS acknowledges the command indicating that the RGS should proceed with the award. This is an important step. It lets the PJS know that the RGS has received the jackpot win notification.

Depending on how the game is designed, there could be a significant delay between the time that the PJS determines that a jackpot has been won and it is actually paid — without this step, the PJS could be left wondering whether the RGS intended to pay the jackpot.

Following the acknowledgement, the RGS pays the mystery jackpot win to the player's account using the `moneyTransactions` command.

After the payment has been made, the RGS reports the final result using the `commitJackpot` command and the PJS acknowledges it.



7.5.1 Mystery Jackpot Notification

The properties used to notify the RGS about a mystery jackpot hit are shown in the table below. Sometimes consolation prizes are paid to other players when a mystery jackpot is won.

Table 7.10 Mystery Jackpot Notification Properties

Property	Description
`controllerId`	Jackpot controller identifier.
`levelId`	Jackpot level identifier.
`referenceId`	Reference identifier; assigned by the PJS.
`hitSeq`	Jackpot sequence number at the time of the hit.
`hitAmt`	Jackpot value at the time of the hit.
`hitText`	Prize description or text representation of the jackpot value at the time of the hit.
`isInKind`	Indicates whether the jackpot is an in-kind prize.
`isConsolation`	Indicates whether the award is a consolation prize.

CHAPTER

8

Content Interface

8.1 Introduction

The content interface is used to coordinate the activities of the iGaming Console (iGC) and the Remote Game Content (RGC). It is used to coordinate player account balances, game state, user interface state, and jackpot values.

TPI specifies which objects should be communicated, not how. The method for communicating the objects is left for suppliers to decide. The objects are specified as JSON objects, but other encodings can be used.

8.2 Player Account Balance

The object used to communicate player account balances through the content interface is identical to the object used to communicate account balances in other parts of TPI. The RGS and RGC should use the same rules for determining the player's available balance and available free spins.

8.3 Game State

The game state object is used by the RGC - the game - to report the current state of the game to the iGC - the console. Often, similar information is displayed in both places.

Table 8.1 Game State Object Properties

Property	Description
`gameReady`	Indicates whether the game is loaded and ready to be played.
`gameInPlay`	Indicates whether the game is currently in play.
`denomId`	Identifies the denomination selected for the game.
`wagers`	Total wagers for the Game Cycle.
`freeSpins`	Total Free Spins used during the Game Cycle.
`wins`	Total wins for the Game Cycle.

The iGC can also request that a game be reloaded or closed.

8.4 User Interface State

Changes to the state of the user interface can be reported by the RGS or the iGC.

Table 8.2 User Interface State Properties

Property	Description
`localeCode`	Indicates the locale (language) being used to communicate with the player.
`mute`	Indicates whether sound has been muted.

These properties can be sent back and forth between the game and the console to keep language and sound coordinated.

8.5 Jackpot Values

The object used to communicate jackpot values through the content interface is identical to the object used to communicate jackpot values in other parts of TPI.

CHAPTER

9

Event Reporting

9.1 Introduction

TPI includes a very flexible event reporting mechanism. The GSA, as well as suppliers, can define events – for example, events to report that a game cycle started or that a game cycle finished.

Events can be reported by the iGP, RGS, or PJS. Events can be reported between the iGP, RGS, and PJS, or events can be reported to external systems – for example, a regulatory monitoring system.

Table 9.1 Event Reporting Properties

Property	Description
`eventId`	Event identifier; unique identifier for the event.
`eventCode`	Event code.
`eventText`	Description of the event.
`eventDateTime`	Date/Time that the event occurred.
Event-Specific Information	
`eventObject`	Identifies the event-specific data object being reported with the event.
`eventData`	Contains the event-specific data object.

The `eventObject` property identifies the event-specific data object being reported with the event. This tells the recipient of the event what to expect to find in the `eventData` property. The `eventData` property contains the event-specific data object.

The set of events reported by an end-point, as well as the destination for the events, are configured through an administrative interface on the iGP, RGS, or PJS. The protocol does not include a subscription mechanism.

The choice of events is based on operational and jurisdictional needs. Different events can be reported to different end-points. The intent is that systems will support the set of events relevant to their role — iGP, RGS, or PJS — and that operators will be able to select which events they want sent to end-points.

From time-to-time, GSA, as well as suppliers, may define additional events to meet operational or jurisdictional needs. When defined by GSA, the events may be included in new releases of the protocol or in technical bulletins.

CHAPTER

10

Triggers

10.1 Introduction

Triggers are used to access special purpose functionality that is not present in TPI but may be needed to meet unique operational or jurisdictional requirements – for example, displaying responsible gaming messages.

Individual triggers are not defined within TPI. Only the mechanism for accessing the triggers is defined. From time-to-time GSA, as well as suppliers, may define triggers to meet special needs. When defined by GSA, the triggers will be announced in technical bulletins.

The properties of triggers are communicated as JSON objects. JSON schemas should be distributed by the creators of triggers.

www.gamingstandards.com