UNLEASH THE POWER OF YOUR FLOOR

The Value G2S® Delivers to an Operator

3rd edition

Unleash the Power of Your Floor, 3rd edition

Released 2018/05/05, by International Gaming Standards Association® (IGSA®).

Second Edition released 2019/10/16.

Third Edition released 2020/03/01.

Patents and Intellectual Property

Trademarks and Copyright

IGSA Contact Information

E-mail: sec@IGSA.org

WWW: http://www.IGSA.org

Table of Contents

About This Document

This document provides a high-level description of the G2S message protocol standard. This document is designed to provide the reader with a conversational overview of G2S. 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 IGSA for information on training materials, online classes and on-site classes.

1.1 Acknowledgements

The International 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 Overview

The International Gaming Standards Association (IGSA) is the de facto source for communication protocols in the gaming industry. IGSA has created an entire suite of communications protocols for gaming operations. Communications protocols for game to system, system to system, and game to peripheral devices all exist. And, IIGSA has created the only standardized communications protocol for online gaming in the industry.

The remainder of this chapter provides a brief overview of the Game-To-System (G2S) protocol, discusses how G2S can stream-line casino floor operations, lists benefits possible by using G2S, and provides EGM transition-to-G2S information.

Chapter 2, G2S vs. SAS provides detailed information highlighting, by functional category, how G2S' capabilities far surpass anything even remotely possible using SAS.

Chapter 3, IGSA Case Studies provides real-world examples showing how the integration of IGSA standards can help operators reduce costs, improve the customer experience, and free up employee resources.

Chapters 4 through Chapter 8 groups G2S classes by functional categories, with descriptions of each class along with command listings providing supported class events and meters.

Appendix A provides information on three central G2S concepts: machine model, connection model, and G2S message handling.

Appendix B provides a high-level description of other IGSA standards and protocols.

1.2 What is G2S?

G2S is designed to be the gaming industry's protocol for network gaming floors. It allows an EGM to communicate directly to a host system over a standard off-the-shelf communications network. In fact, it allows an EGM to communicate with multiple hosts systems simultaneously, each managing different sets of business functionality within the EGM.

This makes G2S different than other protocols. Most other protocols are not designed to communicate directly to a host system. Usually, there is an interface board in between the EGM and the host system, and that interface board handles some of the business functionality. Some of the functionality will be handled by the EGM, and some of it will be handled on the interface board. With G2S, everything is done by the EGM. There are no interface boards. This makes G2S significantly different than other protocols.

G2S is based on existing, proven technologies. G2S leverages as much existing technology as possible. We didn't want to invent something when a solution was already available. XML — Extensible Markup Language — is an excellent example of this. All messages in G2S are encoded in XML.

With G2S v3.0, there is also a provision for using EXI — Efficient XML interchange — a binary version of XML.

G2S specifies the content, format, and processing requirements for messages. Message transport — that is, moving the messages between endpoints — is addressed in a separate set of protocols - the IGSA Transport and Security protocols. Refer to *Appendix B* for more information.

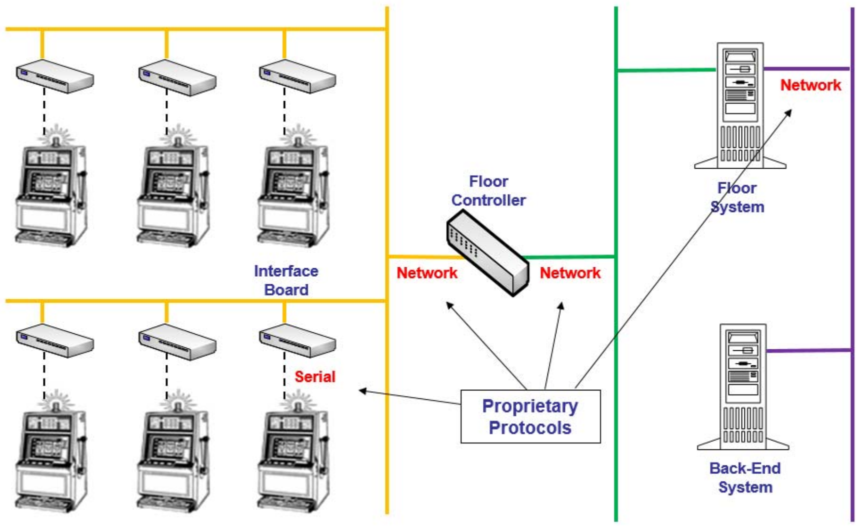
1.3 Why Do We Need G2S?

There are many reasons to incorporate the G2S protocol into your casino operation. SAS, and other legacy protocols, have very limited bandwidth. They are low-speed serial protocols.

The illustration below shows a traditional casino floor topology that does not make use of G2S. Notice how EGMs all communicate using a serial protocol to a slot machine interface board, or SMIB. These SMIBs communicate with a proprietary protocol to a floor controller.

These floor controllers communicate with another proprietary protocol to a central floor system. And finally, the floor systems communicate using yet another proprietary protocol to back-end application systems.

The net result is a whole series of proprietary nodes and protocols layered one upon another. These types of configurations can be very difficult and cumbersome to upgrade when new business functionality is introduced.



Advanced features, such as software download and remote configuration are impractical (or impossible) using such protocols. As demand for such features increased, it became evident that a network-based protocol with significantly higher bandwidth was needed to eliminate the bottlenecks and simplify the topology. Enter G2S.

1.4 Benefits of G2S

Integrating G2S into a casino operation provides a wide range of benefits. These benefits improve operational efficiency in numerous ways. Full adoption and deployment of G2S allows host gaming systems to communicate directly with games and with players. It also allows operators to eliminate unnecessary hardware components and middleware. Operator benefits are highlighted throughout this document. Look for sidebar snippets titled **Benefit**.

Some of the benefits of using G2S include:

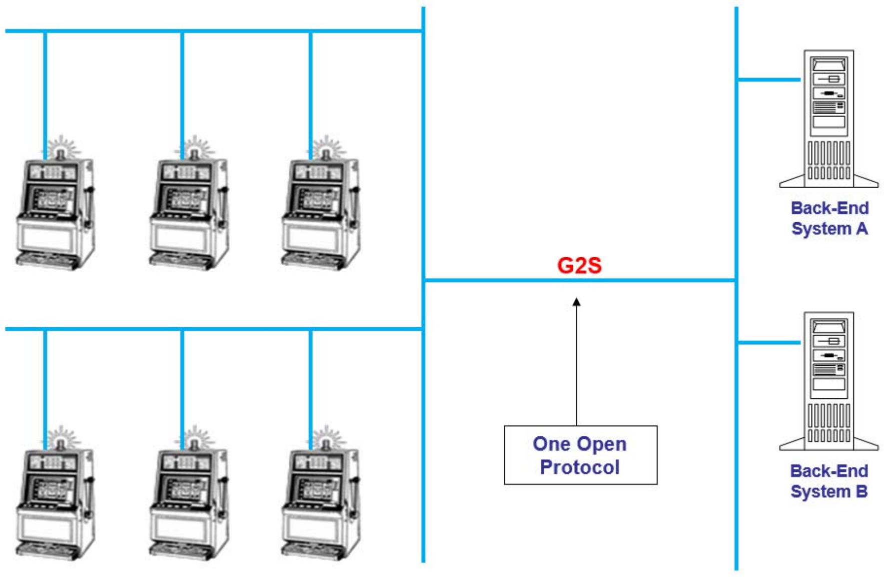
- Direct communications between the EGM and multiple host systems — no SMIBs required.
- Network-based communications using 100Mb/sec TCP/IP connections with TLS 1.2 for security
  - No more serial-based, proprietary protocols
- Descriptor lists provide a complete inventory of the logical and physical devices within the EGM including vendor, product, release, and serial number.
- The meters, status, and logs affected by an event can be sent with the event.
- Meters are always in balance when reported.
- The set of active games and denominations can be configured remotely by a host.
- Game recall log includes detailed information about each game that was played and the win levels that were hit.
- Primary game meters as well as secondary game (double-up) meters are monitored.
- Theoretical win meters are calculated based on the actual wager categories played.
- Mappings between games and progressive jackpots can be configured remotely.
- Mappings between games and mystery jackpots can be configured remotely.
- Standalone progressive jackpot controllers can be monitored and configured remotely.
- Different win levels within a game can be mapped to different jackpot controllers – for example, a wide-area controller and a local-area controller.
- Contribution meters report the qualifying wagers towards jackpots.

- Three types of player bonuses are supported: standard, wager match, and jackpot multiplier.
- The EGM manages the award of wager match and jackpot multiplier bonuses based on parameters set by the host.
- Processor resets and master resets can be triggered remotely.
- Time zone offsets can be configured into the EGM to manage the transition to/from daylight savings time.
- Operating hours can be configured into the EGM to automatically enable/disable the EGM at prescribed times of day.
- Occupancy meters track the amount of time an EGM is actually in use.
- Supports acceptance of foreign currencies; currency exchange rates may be configured remotely.
- Handpays may be keyed off locally by an attendant or remotely by the host.
- Host may select the payment method for a handpay - attendant, ticket, wagering account, smart card, or credit meter.
- Manual authentication codes are printed on tickets to support offline ticket validation.
- Transfers can be initiated by the host or the EGM.
- Smart cards can also trigger player tracking sessions.
- Smart-card transfers can be managed by the EGM through a secure transaction module or via application-specific commands exchanged with a host.
- Funds can be applied to the credit meter or paid to a ticket.
- SMIBs are not needed to track players and employees.
- Door access permissions can be set by the host to control the reporting of legal/illegal door opens.
- The presence of a valid player ID triggers player tracking: wagers, actual win, theoretical win, and other meter movements.
- Points can be calculated using any of the meters captured during a player session; points can also be awarded by a host.
- Three different point calculations can be active – default, player-specific, and generic (time-of-day); the player receives the best award out of the three.
- Up to five hot player levels can be set; when a player reaches a new level, an event is generated.
- Can be used to implement a responsible gaming program.
- PINs can be used to authenticate players.

- The presence of a valid employee ID triggers employee tracking; all meter movements while the employee ID is present are monitored.
- Employees can report activity codes indicating why they visited the EGM.
- Tournaments can be configured and activated remotely.
- Registration fees can be collected via DFT.
- Winnings can be paid via DFT.
- Hosts can define printer templates and then use those templates to print documents – receipts, coupons, promotional offers, etc.
- Content written in HTML or Flash can be loaded, activated, and displayed to players.
- PUI content can communicate directly with the EGM through the EMDI protocol.
- PUI content can communicate with third-party application servers.
- Third-party applications can add/remove funds from the EGM via DFT.
- Protocol-related options and manufacturer-defined options can both be configured remotely.
- Communications between EGMs and hosts can be configured remotely.
- Software packages can be downloaded and installed onto EGMs and their peripheral devices.
- Software installed on the EGM and peripheral devices can be verified remotely.

1.5 G2S Casino Floor — The Concept

The illustration below shows a casino floor topology using G2S. You'll immediately notice there are no layers of proprietary hardware and software as in the prior illustration. This is a much more efficient, lower cost way of operating your casino floor. Refer to *Chapter 3, IGSA Case Studies* for real-world examples of how integrating G2S dramatically improves player experience and efficiency on the casino floor.



1.6 G2S Casino Floor — The Reality

With G2S, front-end systems are needed to manage the traffic coming in from the floor. It is much more efficient to offload generic tasks onto front-end systems and dispatch important messages from front-end systems to back-end systems.

Therefore, two protocols are needed:

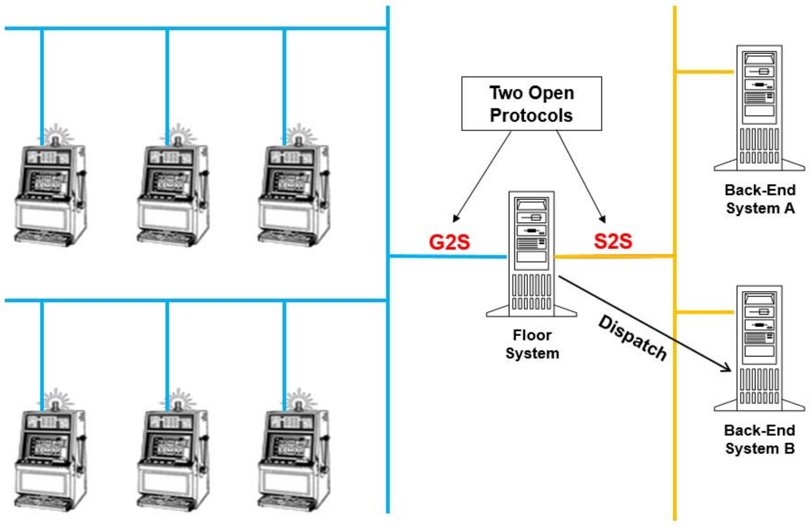
- G2S to communicate between the EGMs and the front-end systems, and
- S2S (IGSA's System-to-System protocol) to dispatch messages from the front-end systems to the various back-end systems.

A series of horizontally-scalable front-end systems manage the traffic. When a front-end system receives something important, it dispatches this to a dedicated back-end system for processing.

The illustration below demonstrates how this topology looks.

Benefit: Once G2S is learned, S2S is easy to learn.

One of the key roles of S2S is to allow G2S messages to be dispatched from generic front-end systems to dedicated back-end systems.



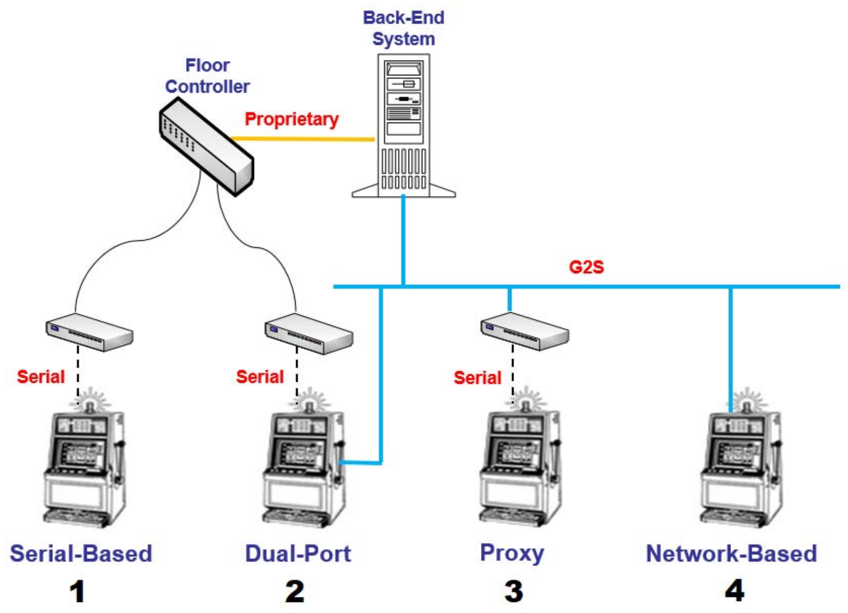
1.7 Transition

Transitioning from proprietary networks to open standards-based networks will take some time. Operators must preserve their investments in their EGMs and transition to the new technology in a logical manner as availability of business resources dictate.

In many cases, it would be cost prohibitive to remove all non-G2S EGMs on a casino floor and replace them with G2S-compliant machines in one fell swoop. More likely, there would be a transition period taking place over a long period of time - perhaps years.

Gradually, older machines would be retired as new machines are introduced. During this transition period, existing machines and newer machines must co-exist. Therefore, hybrid configurations are needed during the transition.

The illustration below shows four EGM configurations that are possible during a transition period.



The left side (#1) displays where non-G2S systems are today: EGMs communicating over a serial communications line to an interface board, up to the floor controller, and eventually to the back-end systems.

The right side (#4) displays a system using G2S: EGMs communicating directly to the back-end system over a network using G2S.

In between are two common transition strategies.

First, dual-port (#2). In this scenario, the G2S protocol is used for advanced functionality not available through the serial protocol. Dual-port opens up advanced G2S functionality without disrupting legacy applications. This is a popular option for download, remote configuration, bonusing, and progressives.

For example, software download and remote configuration might be done through G2S, while player tracking and accounting might still be done using the legacy protocol. Progressives and bonusing could also be done this way.

Note: These capabilities are limited to the capabilities of the underlying serial protocol.

Second, proxy (#3). In this scenario, an interface board translates the serial protocol into G2S. This is a popular scenario for wide-area monitoring. G2S is run over the wide-area connection between the interface board and the back-end system. The serial protocol extracts data from the EGM and G2S hands off that data to wide-area monitoring applications. A proxy-based system allows some G2S capabilities to exist on legacy machines.

Both strategies — dual-port and proxy — are effective in the right environment.

1.8 Getting Started

Getting started with G2S can be a daunting task. There is a lot to learn and do. Fortunately, IGSA and third-party suppliers can make this task easier.

- **Education** — The first step is to learn about G2S and the functionality that will be encountered while implementing G2S. There are a number of resources available from IGSA that can help.
  - Introduction to G2S — This book "Unleash the Power of Your Floor" provides an excellent overview of G2S. It's a great place to start.
  - Online Training — IGSA, in collaboration with iGaming Academy, has created an online lecture series for its standards. This is a great way to learn more about G2S at a minimal cost. See *igacademy.com/partnerships/IGSA* for more details.
  - Personalized Training — IGSA offers personalized onsite training to manufacturers, operators and regulators. This service is available on a time and expenses basis. Contact IGSA staff for more details.
- **Protocol Stacks** — Rather than developing one from scratch, it may be more cost-effective and expedient to purchase a G2S protocol stack from a third-party supplier. This approach can also ease the long-term maintenance burden. Protocol Stacks are available for both EGMs and systems. See the *IGSA website* for more information about suppliers of protocol stacks.
- **Test Tools** — Testing a new implementation of G2S can be as daunting as developing one. Fortunately, off-the-shelf test tools are available from third-party suppliers. These tools can be used to test EGM implementations as well as system implementations. They are great hands-on learning tools as well. See the *IGSA website* for more information about suppliers of test tools.
- **Certification** — IGSA certification is the final proof that an implementation has been done correctly. It demonstrates that the implementation complies with the G2S specification and that it will interoperate with other products that have been certified to the same level, giving operators and regulators additional confidence in the implementation. See the *IGSA website* for more information about certifying your implementation.

CHAPTER

2

G2S vs. SAS

2.1 Introduction

For many years, SAS has been the most widely used protocol in the gaming industry. However, it has limitations: most notably, the need for additional middleware and hardware (SMIBs) to achieve the functionality which is native to G2S.

This chapter contains information demonstrating how G2S stacks up against SAS. The table in the next section summarizes the functional differences between G2S and SAS, grouped by functional category. *Section 2.3, Functionality Description* lists G2S classes, grouped by these functional groups, comparing G2S functionality with SAS functionality.

2.2 Functional Comparison of G2S and SAS

The following table shows the major EGM functionality and which protocol supports it natively. This excludes functionality provided by external, proprietary Slot Machine Interface Boards, (SMIB). Use cases are used to help explain each function in the sections that follow the table.

Functional Description		G2S Native	SAS Native
Communications	Multi-Host Communications Support	Yes	Limited
	Remote Communications Configuration	Yes	No
Meters & Events	Meter Subscriptions	Yes	No
	Persisted Audit Meters	Yes	No
	Event Subscriptions	Yes	No
	Employee Activity Tracking	Yes	No
Game Play, Jackpots & Bonuses	Game Management	Yes	Limited
	Progressive Jackpots	Yes	Limited
	Mystery Jackpots	Yes	Limited
	Stand-Alone Progressive Jackpots	Yes	No
	Server-Based Bonusing	Yes	Limited
	Central Determination	Yes	No
Players	Player Activity Tracking	Yes	No
	Picture-In-Picture Messaging	Yes	No
	Tournaments	Yes	Limited
Money Movements	Handpays	Yes	Limited
	Ticket-In-Ticket-Out	Yes	Limited
	Wagering Account Transfers	Yes	Limited
	Smart Cards	Yes	No
	Direct Funds Transfers	Yes	No
	Forced Cash-Outs	Yes	Limited

Functional Description		G2S Native	SAS Native
Cabinet & Peripheral Devices	Overall Cabinet Management	Yes	Limited
	Coin Acceptors	Yes	Limited
	Hoppers	Yes	Limited
	Note Acceptors	Yes	Limited
	Note Dispensers	Yes	No
	Card Readers	Yes	No
	Printers	Yes	Limited
	Jackpot Signs	Yes	No
	Hardware Component Reporting	Yes	No
	Data Storage Reporting	Yes	No
Remote Configuration	Remote Option Configuration	Yes	No
	Remote Option Configuration by Theme	Yes	No
	Software Download	Yes	No
Regulatory	Software Authentication	Yes	No
	Responsible Gaming Enforcement	Yes	No

2.3 Functionality Description

2.3.1 Communications

Multi-Host Communications (communications Class)

Benefit: Each subsystem on an EGM can interact with the EGM independently of the other systems.

G2S — Is a high-speed (100mb) networked protocol supporting secure TCP/IP communication channels between an EGM and multiple G2S hosts. Support for seven hosts connected simultaneously to an EGM is recommended using point-to-point and multicast communications. Each G2S host may direct the EGM to perform any of the functions it supports. This means that if an Operator has a Casino Management System, a Monitoring and Reporting System, and a Marketing Analytics system, each system can perform functions it supports interacting with the EGM independently of the other systems.

SAS — Is a low-speed (19.2kb) serial protocol, communicating through a serial port to a single host. Two serial ports may be available on some EGMs and, therefore, it is possible to have two hosts connected. However, it is highly recommended that only one SAS host (Primary) direct the EGM to perform critical functions while the other host (Secondary) have read-only access. This is to avoid conflicting instructions from the different hosts since SAS has no idea that multiple hosts exist.

Remote Communications Configuration (commConfig Class)

Benefit: Remote configuration is possible for each host using the exact functionality needed by the application.

G2S — Used to identify the hosts with which an EGM should communicate and to set the device access permissions for those hosts. This means that if an Operator has a Casino Management System, a Monitoring and Reporting System, and a Marketing Analytics system, each can be configured remotely to communicate with the EGM using the exact functionality needed by the application.

SAS — Not possible through the SAS protocol.

2.3.2 Meters and Events

Meter Subscriptions (meters Class)

Benefit: EOD, periodic, drop, door open, snapshot, etc. meters can be collected for every game in a multi-game set.

G2S — Used to set meter subscriptions (end-of-day, periodic, drop, door open, snapshot, etc.). Meters are sent when the associated trigger points occur. Polling is also possible. Meters can be collected for every game in a multi-game set, allowing the Operator to determine which games are the most popular / profitable.

SAS — Not possible through the SAS protocol. The EGM must always be polled for meters.

Persisted Audit Meters (auditMeters Class)

Benefit: EOD meters can be read by any host system.

G2S — Used to persist a set of end-of-day meters which can be read by any host system. The meters are persisted until overwritten at the next end-of-day.

SAS — Not possible through the SAS protocol.

Event Subscriptions (eventHandler Class)

Benefit: subscriptions for events and associated data can be sent to the desired system.

G2S — Used to set subscriptions for events and associated data (meters, device status, and transactions). Associated data is sent with the events when they occur. The operator selects which events are sent to the different systems connected to the EGM. The subscription process ensures that only the data needed by a system is sent to it.

SAS — Not possible through the SAS protocol.

Employee Activity Reporting (employee Class)

Benefit: G2S can isolate employee-caused changes in meters from normal game activity.

G2S — Used to track meter movements while an employee is present at an EGM, allowing casino accounting to exclude meter movement due to employee preventative maintenance or EGM testing, from normal game activity. Can also be used to report activity codes entered by an employee while at an EGM, allowing automation of machine entry access logs.

SAS — Not possible through the SAS protocol.

2.3.3 Game Play, Jackpots & Bonuses

Game Management (gamePlay Class)

Benefit: With G2S, you have access to game recall and outcome logs.

G2S — Used to enable/disable individual games and denominations, allowing the operator to enable individual games within a multi-game set and to select the denominations to offer to the player for each game. Provides access to game recall and outcome logs.

SAS — Game recall and outcome logs are largely not supported even with the most recent enhancements.

Progressive Jackpots (progressive Class)

Benefit: Can work with multiple independent jackpot controllers and levels. Reconciliations can be done using G2S data collected remotely via the CMS or a separate G2S Host.

G2S — Used to manage the payment of progressive jackpots. Supports multiple independent jackpot controllers and levels. Different games can be linked to different controllers providing unlimited configuration possibilities. Jackpots can be paid to the credit meter, by handpays, by vouchers, or to wagering accounts. Contribution meters help simplify controller implementations and reconciliations. Reconciliations can be done using G2S data collected remotely via the CMS or a separate G2S Host, without having to manually read progressive jackpot meters on the casino floor.

SAS — Remote configuration, contribution meters, and payments to the credit meter, by vouchers, and to wagering accounts are not supported.

Mystery Jackpots (mystery Class)

Benefit: Different games and denominations can be linked to different controllers providing unlimited configuration possibilities.

G2S — Used to manage the award of mystery jackpots. Supports multiple independent jackpot controllers and levels. Different games and denominations can be linked to different controllers providing unlimited configuration possibilities. Jackpots can be paid to the credit meter, by handpays, by vouchers, or to wagering accounts. Contribution meters help simplify controller implementations and reconciliations. Reconciliations can be done using G2S data without having to manually read mystery jackpot meters on the casino floor.

SAS — Remote configuration, contribution meters, and payments to the credit meter, by vouchers, and to wagering accounts are not supported. Limited to one external mystery jackpot connected to the secondary communications port.

Stand-Alone Progressive Jackpots (spc Class)

Benefit: Remotely configure internal stand-alone progressive jackpots within an EGM.

G2S — Used to remotely configure internal stand-alone progressive jackpots within an EGM and to report current jackpot values and resets.

SAS — Not possible through the SAS protocol.

Server-Based Bonusing (bonus Class)

G2S — Used to award server-determined bonuses to players. Wager-match and jackpot multiplier bonuses are managed by the EGM using parameters set by the server. Awards can be paid to the credit meter, by handpays, to vouchers, or to wagering accounts.

SAS — EGM-managed wager-match bonuses and payments to vouchers and wagering accounts are not supported.

Central Determination (central Class)

Benefit: Allows an EGM to request game outcomes from a central determination system. (for Class II gaming)

G2S — Used by an EGM to request game outcomes from a central determination system. Specifically designed to meet the needs of Class II gaming in the United States.

SAS — Not possible through the SAS protocol.

2.3.4 Players

Player Tracking (player Class)

Benefit: Can be used to award points and report hot players.

G2S — Used to track play while a player is present at an EGM including win/loss, time played, and theoretical win/loss. Can be used to award points and report hot players. Time-based and player-specific point calculation overrides are available.

SAS — Not possible through the SAS protocol.

Picture-In-Picture Messaging (mediaDisplay Class)

Benefit: PUI content can communicate directly with the EGM through the EMDI protocol.

G2S — Used to manage the Player User Interface – picture-in-picture windows through which players can interact with third-party applications on the main screen or secondary screens of the EGM.

SAS — Not possible through the SAS protocol.

Tournaments (tournament Class)

Benefit: Manage tournaments; enroll players, initiate sessions, report results.

G2S — Used to manage slot tournaments - enroll players, initiate sessions, report results. Supports embedded EGM-based user interfaces as well as external host-controlled user interfaces.

SAS — Can only be used to initiate tournament sessions.

2.3.5 Money Movements

Handpays (handpay Class)

Benefit: Handpays may be keyed off locally by an attendant or remotely by the host.

G2S — Used to report large wins and cancel credits that exceed configurable limits. Supports local key-offs by attendants, remote key-offs by systems, as well as key-offs to the credit meter, vouchers, and wagering accounts.

SAS — SAS does not support remote key-offs by systems but allows manual attendant key-offs to the credit meter, vouchers and wagering accounts.

Ticket-In-Ticket-Out (voucher Class)

Benefit: Can be used to manage the issuance and redemption of cash-out and promotional tickets.

G2S — Used to manage the issuance and redemption of cash-out and promotional tickets. EGM may request multiple validation Ids to support offline voucher issuance. Includes a method for validating tickets that were printed while an EGM was offline. Vouchers tied to players when issued and redeemed.

SAS — Does not allow an EGM to request multiple validation Ids. Does not include a method for validating tickets that were printed while an EGM was offline. No player information.

Wagering Account Transfers (wat Class)

Benefit: Transfers can be initiated by the host or the EGM.

G2S — Used to transfer funds to/from player accounts on a host system. Supports embedded EGM-based user interfaces as well as external host-controlled user interfaces.

SAS — Embedded user interfaces are not supported.

Smart Cards (smartcard Class)

Benefit: Smart card transfers can be managed by the EGM through a secure transaction module.

G2S — Used to manage and report transactions associated with smart cards. Allows application data to be relayed to host systems and verified. Includes secure transaction module status information.

SAS — Not possible through the SAS protocol.

Direct Funds Transfers (dft Class)

Benefit: Can be used to transfer funds between an EGM and a host system application.

G2S — Used to transfer funds between an EGM and a host system application. Designed to be used in conjunction with third-party applications running in Player User Interface windows - for example, to purchase keno tickets or pay tournament fees.

SAS — Limited functionality through the SAS protocol using AFT commands.

Forced Cash-Outs (cashout Class)

G2S — Used to remotely initiate a full or partial cash-out from an EGM.

SAS — Limited functionality through the SAS protocol using the AFT command.

2.3.6 Cabinet & Peripheral Devices

Overall Cabinet Management (cabinet Class)

G2S — Used to report the overall status of the EGM including door status, tilts, language, last game played, selected game, etc. Can be used to enable/disable entire the EGM, or selectively enable/disable game play or money in. Supports remote processor reset, remote master reset, operating hours, and time zone changes.

SAS — Features such as language, remote resets, operating hours, and time zone changes are not supported.

Coin Acceptors (coinAcceptor Class)

Benefit: Supports multiple currencies and configurable exchange rates.

G2S — Used to report activity associated with coin acceptors, such as tilts, faults, and drop door access. Supports multiple currencies as well as configurable exchange rates.

SAS — Multiple currencies and configurable exchange rates are not supported.

Hoppers (hopper Class)

Benefit: Supports multiple currencies and configurable exchange rates.

G2S — Used to report activity associated with hoppers, such as tilts, faults, hopper door access, and hopper status (full, empty, high-water mark, etc.) Supports multiple currencies as well as configurable exchange rates.

SAS — Multiple currencies and configurable exchange rates are not supported.

Note Acceptors (noteAcceptor Class)

Benefit: G2S supports multiple currencies as well as configurable exchange rates.

G2S — Used to report activity associated with note acceptors, such as tilts, faults, and stacker door access. Supports multiple currencies as well as configurable exchange rates.

SAS — Multiple currencies and configurable exchange rates are not supported.

Note Dispensers (noteDispenser Class)

Benefit: Supports multiple currencies and configurable exchange rates.

G2S — Used to report activity associated with note dispensers, such as tilts, faults, dispenser door access, and dispenser status (full, empty, high-water mark, etc.) Supports multiple currencies as well as configurable exchange rates.

SAS — Not possible through the SAS protocol.

Card Readers (idReader Class)

G2S — Used to validate IDs presented at an EGM. Supports multiple types of ID readers including magnetic cards, RFID cards, and biometric scanners. Allows host systems to see which player or employee is present at an EGM. Allows card readers to be connected directly to the EGM's mother board as peripheral devices. This eliminates the need for the separate player tracking hardware used in legacy EGMs.

SAS — Not possible through the SAS protocol.

Printers (printer Class)

Benefit: Hosts can define printer templates and then use those templates to print documents - receipts, coupons, promotional offers, etc.

G2S — Used to report activity associated with printers, such as tilts and faults. Can be used to configure templates into the printer and perform remote printing of receipts, coupons, and promotional items from those templates.

SAS — Remote printing is not possible through the SAS protocol.

Jackpot Signs (sign Class)

Benefit: Configure and control the display of jackpot information on signs managed by an EGM.

G2S — Used to configure and control the display of jackpot information on signs managed by an EGM. Used in conjunction with progressive and mystery jackpots.

SAS — Not possible through the SAS protocol.

Hardware Component Reporting (hardware Class)

Benefit: Remotely gather a complete list of hardware components, including model, serial number and firmware versions.

G2S — Used to report the list of hardware components installed within an EGM as well as the capabilities of those components. Allows the operator to remotely gather a complete list of hardware components, including model, serial number, and the software / firmware versions they are running. An incredibly useful tool when dealing with revoked or obsoleted software / firmware.

SAS — Not possible through the SAS protocol.

Data Storage Reporting (storage Class)

G2S — Used to determine the amount of storage available on an EGM for storing and installing software packages.

SAS — Not possible through the SAS protocol.

2.3.7 Remote Configuration

Remote Option Configuration (optionConfig Class)

Benefit: With G2S, active games and denominations can be configured remotely by a host.

G2S — Used to remotely configure protocol-related and manufacturer-specific options for an EGM, allowing the Operator to manage the configuration of EGMs from a central location.

SAS — Not possible through the SAS protocol. Configuration must be done locally at the EGM.

Remote Option Configuration by Theme (gameTheme Class)

G2S — Used to configure theme-related options that are shared amongst multiple games within an EGM.

SAS — Not possible through the SAS protocol. Configuration must be done locally at the EGM.

Software Download (download Class)

Benefit: With G2S, software packages can be downloaded and installed onto EGMs and their peripheral devices.

G2S — Used to download and install software on an EGM and its peripheral devices, such as printers and note acceptors, allowing the Operator to manage the software/firmware installed on EGMs from a central location.

SAS — Not possible through the SAS protocol. Software/firmware must be installed locally at the EGM.

2.3.8 Regulatory

Software Authentication (gat Class)

Benefit: G2S allows a complete inventory of the logical and physical devices within the EGM including vendor, product, release, and serial number.

G2S — Used to request the inventory of software/firmware components installed on an EGM and its peripheral devices and then to calculate the signatures of those components. Supports a wide range of algorithms, including SHA1 and SHA2. Fully compatible with the serial GAT protocol. Allows the Regulatory Agent to request GAT signatures remotely from a central location rather than connecting a laptop to each individual EGM on the casino floor, reducing EGM downtime and operational impact.

SAS — The SAS protocol only supports a limited subset of the GAT protocol.

Responsible Gaming Enforcement (informedPlayer Class)

G2S — Used to access various controls available in the EGM to promote responsible gaming including game speed, maximum bet, continuous play, PIN activation, etc.

SAS — Not possible through the SAS protocol.

CHAPTER 3 IGSA Case Studies

3.1 Introduction

This chapter presents examples of how G2S is being used in the real world. Additionally, included here are reprinted articles from major industry publications which discuss the compelling reasons the G2S protocol should be integrated into your casino operation. Sidebar snippets are added pointing out important information.

In this chapter, you'll find:

- *Section 3.2, G2S Enables Casino to Develop Unique Product*. How a subsidiary of Loto-Québec achieved results beyond their expectations
- *Section 3.3, G2S Protocol Plays Key Role in the Intralot iGEM™ Monitoring System*: Three examples of how G2S became the central component of casino operations
  - *Section 3.3.1, Ohio Uses the G2S Protocol to Monitor 10,500 EGMs in 7 Racinos*
  - *Section 3.3.2, Victoria, Australia Monitoring System for Non-Casino Gaming*
  - *Section 3.3.3, Victoria, Australia Pre-Commitment System for Gaming Patrons*
- *Section 3.4, Lessons Learned, and Value Created From Implementing IGSA Standards.* This case study contains the viewpoints

of operators and regulators as regards the effect IGSA standards have on operations as well as value derived therein.

- *Section 3.5, Bridging the G2S Gap*. A white paper on G2S viability in the distributed gaming market.
- *Section 3.6, Case Study for the Value of IGSA Protocol Implementation*. A case study covering Canadian gaming jurisdictions and their experience integrating IGSA protocols.
- *Section 3.7, Case Study - The Inconvenient Truth - of G2S Adoption*. An analysis of why G2S adoption has not been as widely adopted as expected.

3.2 G2S Enables Casino to Develop Unique Product

(originally printed in 2012)

About three years ago, the Société des casino du Québec (SCQ), a subsidiary of Loto-Québec put forth a vision for the future of its casinos. Integral to that vision was the desire to create a more immersive gaming experience for players. “We really wanted to create an environment where players would see more winning, and the game could become part of the atmosphere within the casino,” said Marie-Josée Parent, manager of new product development for the SCQ.

SCQ approached WMS, and the challenge consisted of taking an existing slot machine game, Jackpot Explosion, and projecting winning jackpots on giant screens.

On June 22, 2012, Volcan was launched at Casino Charlevoix in La Malbaie, Québec. The challenge consisted of taking an existing slot machine game, WMS’ Jackpot Explosion®, and projecting winning jackpots onto giant screens. In all, 28 slot machines with 14 new themes are connected to each other and to a multimedia installation set up in 2010 by Moment Factory. As the jackpot grows, the volcano’s lava level rises and rumbling begins, fueling player excitement.

When a player wins one of the top jackpots, the volcano erupts, producing spectacular visual effects on a 27-screen video wall and four video projection surfaces on the ceiling of the main gaming area. SCQ Project Manager Marc Santerre enlisted the help of the SCQ multimedia team and Ingenio, Loto-Québec's R&D arm, to create the unique, immersive experience.

It was no simple feat, and a key issue was how to get real-time information from the game to the system to signal changes in the multimedia display, such as rising lava levels.

The solution?

International Gaming Standards Association's Game-to-System (G2S) protocol. "We needed the information from the game to figure out how to connect to the environment," said Valérie Messier, project manager, Ingenio. "The most important issue for this project was to take the game inside the machine and bring it out to the multimedia experience, to the whole ceiling of the casino in that area, to see where the progressive was, in what state and in what amount, and to see which machine has won."

Ingenio discussed the issue with WMS, which suggested using the G2S protocol and that Ingenio might want to enlist Reno-based Radical Blue Gaming to develop a G2S engine application to facilitate the plans.

Benefit: Without G2S, 'it would have been a lot harder to do this."
- Valérie Messier
Project Manager, Ingenio

With Radical Blue's product and an application from Ingenio merged to a visual recognition technology to analyze the information and trigger the system in the casino, the problem was solved. Without G2S, "it would have been a lot harder to do this," Messier said. "There were some solutions that we thought about, but this was the easiest possible solution."

As the first tool created by an operator using G2S, it stands as a landmark achievement. "Casino de Charlevoix wanted to create a promotion with advanced multimedia presentations that would immediately and dynamically communicate directly to customers throughout the casino on large display monitors," said Russ Ristine, Radical Blue president.

"They needed information from existing WMS slot machines to make the promotion work, so we helped them tap into the existing G2S port on the game, created a simple engine application that connected to their new promotion application, and in less than 90 days, their goal was realized."

The application empowers Casino de Charlevoix to communicate directly to customers with a wide range of messages — from jackpots to available seating — all designed to generate excitement and drive traffic.

Benefit: "...we're getting fantastic comments from our customers who are really enjoying the immersive state."
- Marie-Josée Parent
Manager of New Product Development for SCQ

SCQ and Ingenio couldn't be more pleased with the results, Parent said. "This is one of the first products from our vision that has been implemented, and we're getting fantastic comments from our customers who are really enjoying the immersive state." Seats at the games are the first to fill when the casino opens and the last to empty at closing time, she said.

Parent noted the casino really wanted to do something new for its customers. "Obviously our customers go to other casinos, and we wanted to create that immersive environment that they don't see when they visit other venues."

Volcan was introduced during the high summer season. "When the summer season ends and it becomes much calmer, we are looking forward to see how the customers react and if it's still the hottest game and so forth," Parent said, noting the games generate much more than house average. In fact, after only 75 days, the 28 units, which represents 3 percent of the slot floor, generated an 11 percent lift.

Benefit: "Usually when customers play a game and they win, no one knows. Now it's like a social event. They're celebrating together."
- Marie-Josée Parent
Manager of New Product Development for SCQ

Volcan also has created a new social element in the casino. "Usually when customers play a game and they win, no one knows. Now it's like a social event. They're celebrating together." The whole multimedia experience ups the entertainment quotient throughout the casino, Parent noted. "You have the trembling of the volcano when it starts to rise, and it's not just the image, it's also the audio effects, that create excitement among the players."

It's only the beginning. "For us, it's a stepping stone to move forward," she said, noting plans are in the works to go live before the end of the year with similar immersive experiences at two sister casinos in Québec, Casino de Montréal and Casino du Lac-Leamy. The landmark use of G2S is getting noticed, as it produces tangible results and benefits, Santerre said. "I think it's a big step for the industry."

Benefit: "The interesting thing about G2S is that once the game supports the protocol, it becomes easy to write applications for a casino or even to have the casino write the applications that work with the games directly."
- Russ Ristine
President
Radblue Gaming

In the past, standards have been something that the industry talks a lot about. "We see them on paper, but we don't necessarily see it happening," he said. "The great satisfaction at Loto-Québec is that we're making it happen. We're very proud of that." G2S is key to helping casino operators and gaming developers unlock innovation.

In a few years, he predicted, the Global Gaming Expo may offer some very different booths next to traditional vendors. "Where we see the industry going is that maybe three to five years from now, an operator will be able to walk down the aisles and see different types of applications that they can try out in their casinos."

3.3 G2S Protocol Plays Key Role in the Intralot iGEM™ Monitoring System

(originally printed in June 2015)

Intralot's iGEM Monitoring System has been developed based on IGSA standards, making G2S the default communication protocol between the system and the connected EGMs. For non-G2S machines, iGEM provides a protocol converter (SMIB), which is installed in each machine and translates its native protocol to G2S.

Intralot's latest iGEM system implementations in Ohio (USA) and Victoria (Australia) use this concept, making these states IGSA ready.

In both jurisdictions, Intralot has also developed S2S links with 3rd party external systems. Many of the G2S classes and the SMIB conversion software have been certified by IGSA while the S2S links are in the process of submission for approval.

Intralot's three business cases are presented in the following pages.

3.3.1 Ohio Uses the G2S Protocol to Monitor 10,500 EGMs in 7 Racinos

Benefit: "The system Intralot has installed, to monitor and control the 10,500 EGMs in the 7 Ohio Racinos, saves thousands of hours of manual audits and game checks each year alone."
- Matt Johnson
Technical Director
Intralot USA

Intralot, the technology provider to the Ohio Lottery, operates its iGEM Gaming System to monitor and control the State's gaming machines in Racinos statewide. The system monitors more than 10,000 EGMs in 7 locations. It provides native G2S support while

Intralot's iGEM protocol converter offers compatibility to EGMs that still use the SAS protocol.

The Intralot monitoring project started in 2011 as a result of a contract option selected by the Ohio Lottery. The primary goal of the project was to eliminate manual check and auditing of EGMs operating in the seven locations and the policy affects all sixteen EGM suppliers that are licensed by the state to place machines in Racinos.

In addition to the Game Authentication function within G2S (GAT), the monitoring system also allows remote verification of game percentages and paytables[1]. The Intralot solution in Ohio Racinos also takes advantage of multi-host capabilities within machines which is important because Intralot's iGEM System can operate simultaneously, in parallel to the casino management systems used for other daily tasks.

3.3.2 Victoria, Australia Monitoring System for Non-Casino Gaming

Benefit: "The IGSA protocols are not only foundational to our products, but they have already become the global standard that our Lottery customers expect and demand."
- Theodosios Engelis
Group Director
Intralot

In 2011, Intralot was chosen by the state of Victoria in Australia to develop and operate the Central Monitoring and Control System for more than 26,500 EGMs widely distributed throughout the country's most densely populated state. The system was required to control, authorize and monitor all EGM's operating outside casino environments and was fully deployed in February 2013.

Intralot's field-tested iGEM platform was adapted to comply with the requirements in Victoria and also to support other gaming protocols currently used in Victoria, such as QCOM and two versions of the legacy VLC protocol. For that purpose, the iGEM SMIB was employed to provide protocol conversion from native EGM protocols to the IGSA G2S classes and commands that the iGEM Site Controller supports.

To facilitate the iGEM monitoring system requirements in Victoria, Intralot employed the following G2S protocol classes: *communications*, *gamePlay*, *voucher*, *cabinet*, *optionConfig*, *handpay*, *eventHan-*

1. This application link provides support for the task of game's certifications tracking. More specifically enables the communication between Intralot's lottery system and GLI in order to obtain information regarding the approval status of different items that make up a software set. At the same time we provide to GLI cloud info for the Racinos (`propertyInfo` class) and their VLT's (`clientInfo` class).

   Note: the `swSet` class belongs to Intralot's extension, agreed only with GLI.

dler, *gat*, *progressive*, *meters*, *noteAcceptor*, *bonus*.

3.3.3 Victoria, Australia Pre-Commitment System for Gaming Patrons

Following the successful implementation of the Victoria Monitoring System (see *Section 3.3.2*), Intralot was also chosen by the State of Victoria to implement a Central State-Wide Pre-Commitment System enabling players to issue and use cards which allow them to set multiple limits (time and money loss) on their game play. This central system includes a state wide registered-player repository including player-limit and balance information.

To facilitate the pre-commitment requirement, the following G2S classes were used: *gamePlay*, *idReader*, *player*, *informedPlayer* (with Intralot extension).

3.3.3.1 IGSA's System-to-System (S2S) Links with Crown Casino

Crown Casino in Melbourne will maintain their existing systems for player registration and player tracking while a real-time IGSA S2S link has been developed connecting them to Intralot's Pre-Commitment system, to exchange player details, Responsible Gaming parameters as well as Player Session data.

This S2S link will be used for the communication between the Pre-Commitment Central System and the Crown System for the support of the player's sessions at Crown's EGMs, in order to monitor player gaming activity and check against their limits, implementing the selected Pre-commitment scheme across Victoria.

Intralot designed and implemented a specific extension for "informedPlayer" class of the S2S protocol in order to enable two-way exchange of player-session information required for the purposes of linking the Victoria system and the S2S link with Crown Casino systems. This extension has been discussed and agreed with IGSA and is in the process of submission for approval to the S2S committee of the International Gaming Standards Association.

3.4 Lessons Learned, and Value Created From Implementing IGSA Standards

(based on interviews with operators and regulators between February and June 2018)

3.4.1 Objective

In their 2018 strategic plan, the IGSA Board of Directors directed staff to learn about the adoption of IGSA standards, value derived and impediments. This case study represents their findings.

3.4.2 Participants

The IGSA met with North American regulators and operators who volunteered to share their insights and real-world experience on their journey to adopt IGSA standards.

3.4.3 Impact of Gaming Standards

This case study reveals the intended vision, impediments to implementation to be aware of, and the real, measurable value and benefits achieved. We thank all those who participated in this process for their time and expertise and hope it can help others that are on their IGSA standards journey.

3.4.4 Methodology

The meetings were held either in-person or via teleconference and the participants were provided with a series of questions in advance. The questions were open-ended and geared towards generating dialogue. Detailed notes were taken during each meeting.

The result is this case study that presents the benefits and the value-add the standards provide as identified by the interviewees. It further articulates their original vision and goals and identifies the impediments experienced as they sought to implement their vision.

The case study is comprised of Operator and Regulator sections. Each consists of three parts: Vision, Finding, and Conclusion. The last section represents an overarching set of conclusions the IGSA staff derived from the information shared with them.

3.4.5 Executive Summary

The Operators and Regulators interviewed clearly and unequivocally communicated that their use of the standards created by the IGSA have added significant value to their companies and jurisdictions.

By gaining access to new data and driving increased levels of operational efficiencies to enhancing gaming software integrity, IGSA standards provide a clear return on investment.

Those aware of how widely IGSA standards have been adopted across the world agree that the gaming industry in the United States would also benefit from this.

Benefit: “We have never been as stable as we are now with the implementation of the Game to System (G2S) protocol.” - Byron Bridger, AGLC

Users of IGSA standards are taking advantage of the functionality a new and extensible protocol provides versus the limited capabilities of an antiquated and near impossible to enhance protocol such as SAS and other old-tech protocols.

These users have indicated their support for ongoing IGSA standards deployment and for the creation and use of a certification program to ensure uniformity across all implementations. They likewise have asked for curbing so called ‘Private’ or ‘Proprietary’ extensions that dilute the standards by effectively creating competing versions.

Lastly, both operators and regulators are realizing that the most efficient way for IGSA standards to be more broadly adopted within the United States is to have regulatory authorities mandate them, just like other products and processes have been mandated in the past.

This idea of regulatory mandate appeared to be equivalent to regulatory overreach for some. Some were also not aware that many of the gaming machines on casino floors in the United States had IGSA standards within them. Why should operators and regulators in other countries benefit from these technologies while those in the United States lag behind?

A mandate to use IGSA standards need not put untenable burdens on operators, rather they can start small by requiring coexistence of protocols, such as SAS and G2S, in the same gaming device. This would provide added value to Operators without having to change their slot accounting system. Regulators would get value from being able to connect regulatory reporting systems to those same gaming devices without impacting the slot accounting system.

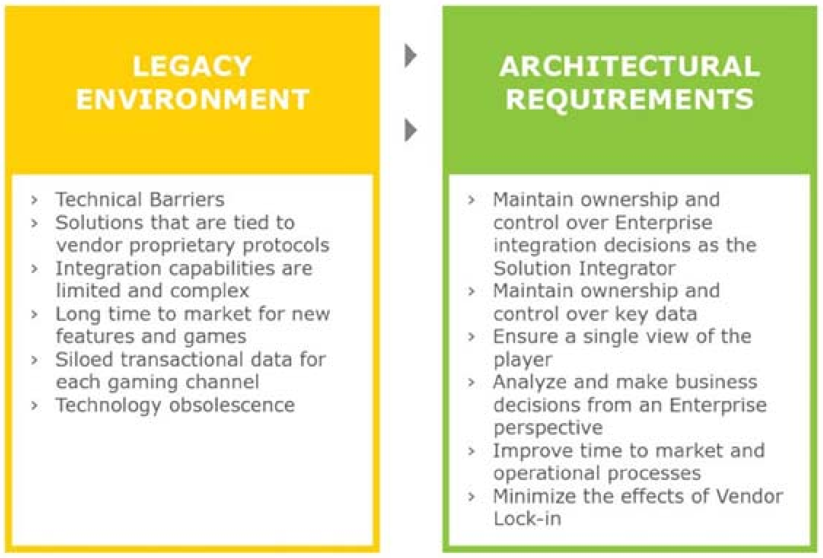
In summary, the standards work and add tremendous value while greatly improving efficiency. Read on to find out more on the value and how you can benefit.

3.4.6 Operator Input

3.4.6.1 Operator Vision

Ten years ago, Canadian operators gathered to discuss their vision for their future operations: *A long-term transformational initiative intended to evolve a gaming Service Oriented Architecture (SOA) through system enhancement, acquisition, and integration and development*.

The group decided that the International Gaming Standards Association would be the best vehicle with which to achieve their goals.


LEGACY ENVIRONMENT
› Technical Barriers
› Solutions that are tied to vendor proprietary protocols
› Integration capabilities are limited and complex
› Long time to market for new features and games
› Siloed transactional data for each gaming channel
› Technology obsolescence
ARCHITECTURAL REQUIREMENTS
› Maintain ownership and control over Enterprise integration decisions as the Solution Integrator
› Maintain ownership and control over key data
› Ensure a single view of the player
› Analyze and make business decisions from an Enterprise perspective
› Improve time to market and operational processes
› Minimize the effects of Vendor Lock-in

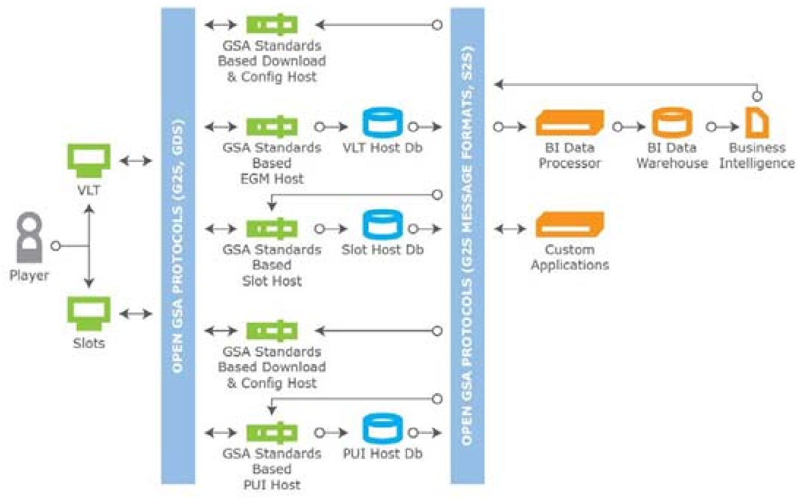
Technology Priorities Identified by Operators

- Support business needs across multiple gaming channels providing improved analytics
- Improve operational efficiencies through increasing business agility and reducing time to market
- Improve relationship management by better understanding the customer

Building the Foundation for Business Intelligence

Delivering the right game, in the right place at the right time

- To position the enterprise for future growth by:
  - Ensuring seamless systems integration through standardization
  - Increasing the agility to respond to market demands
- To improve products analytics across gaming streams by:
  - Enhancing the ability to make better product purchasing and placement decisions
  - Being able to create comparisons between multiple gaming channels
  - Obtaining better insight to the financial performance of the network
- To obtain a single view of the customer by:
  - Offering a seamless and consistent experience to the customer
  - Implementing consistent social responsible programs
  - Offering products to the customer in the right place


GSA Standards Based Download & Config Host
GSA Standards Based EGM Host
VLT Host Db
GSA Standards Based Slot Host
Slot Host Db
GSA Standards Based Download & Config Host
GSA Standards Based PUI Host
PUI Host Db
VLT
Player
Slots
OPEN GSA PROTOCOLS (G2S, GDS)
OPEN GSA PROTOCOLS (G2S MESSAGE FORMATS, S2S)
BI Data Processor
BI Data Warehouse
Business Intelligence
Custom Applications

3.4.7 Operator Findings

Benefit: Operators confirmed that IGSA standards are providing an extremely compelling value proposition that is far exceeding any of their initial expectations.

IGSA Standards Benefits and Value

- The IGSA standards:
  - Have been successfully implemented. The technology is solid.
  - Provide traceability due to the level of data transparency from EGM to System
  - Have made a tremendous amount of data available that we had not been able to take advantage of before. We now have access to slot data that was not possible using SAS. We can pull this data and provide it to the Slot Analysts in a format that enables them to utilize their tools to identify potential changes needed to increase slot revenues.
  - Have provided us a significantly more stable operating environment both at the System (Host) and EGM OS levels. We have never been as stable as we are now with the implementation of the Game to System (G2S) protocol.
  - Have resulted in faster time to market, higher operational quality and optimization of staff levels.
  - Allow us to update marketing and promotional messages to all EGMs quickly using the Player User Interface (PUI). This provides our Marketing team with the flexibility needed to attract and retain players through engaging and fresh content.
  - Allow the Slot Operations staff to process a Jackpot in seconds instead of minutes using G2S's Jackpot tax W2G report accrual functionality. This not only makes the winning experience better for the players, it allows them to get back in the action faster providing significant value.

- The IGSA standards have by far exceeded expectations in a variety of different areas:
  - The level of operational efficiency resulting from implementing this protocol versus the previous way things were done using the older protocol, has led to a multi-million dollar savings for our organization.

  - The standards have significantly enhanced our capabilities to manage our business and have provided the awareness we were looking for.
  - The standards have provided us way more than we thought they could ever offer us.

  - The standards have opened up opportunities for future growth we could not even envision.
  - The G2S standard allows us to utilize and apply the same IT tools and processes to manage the slot floor as we use to manage the enterprise network. This reduces risk, increases uptime, and lets Slot Operations focus on maximizing revenue while IT looks after the infrastructure.

  - With the IGSA standards, we can remotely log into the network, troubleshoot individual EGMs and take corrective action to bring that EGM back online. This functionality enables authorized Slot Operations personnel to resolve issues within 5 to 10 minutes instead of hours.

An extremely compelling value proposition for the casino operator is the ability to manage many EGM administrative tasks from downloading Operating System (OS) and Peripheral device code to adding other customer-value services.

3.4.7.1 Operator Recognition of the Critical Importance of IGSA Certification

Not insisting on IGSA Certification resulted in:

- Initial integration challenges between Host and EGM providers due to the variety of interpretations on how to implement the standard. To revolve this:
  - We were forced to create our own ‘how to’ guideline that documents how a vendor should build a platform for this market, and how to implement the various classes and messages.
  - The device providers take our ‘how to’ guidelines and build to that spec. As a result we don’t have to act as an integrator anymore. This leads to almost seamless integrations.

What would we have done differently?

- We should have forced the manufacturers to get together and figure out product integration without us having to be the middleman.
- We should have asked IGSA to be more actively involved in the integration testing.
- We all should have agreed on a gold standard. IGSA Certification is the critical requirement to get to inter-operable solutions. Our inability to insist on IGSA certification, led to both short-term and long-term pains.

3.4.7.2 Operator Observation

- Our Host supplier single handedly took a position on any ambiguity in the standard and decided how it should be implemented. IGSA was not consulted.
- Today Host suppliers still insist that EGM suppliers sign NDAs before the Host suppliers share their book of ‘trade secrets’ or the parameters that are unique to their system

on how to implement the standard as they interpreted and defined it.

- Supplier extensions to the G2S standard which are protected via NDAs are diluting the value of the standard. The extensions create new proprietary protocol versions. So instead of having one standard we have multiple unique versions. Proprietary extensions should be part of the open standards.
- Few Slot Operators involve IT and Marketing in purchasing decisions. This perpetuates the status quo, i.e. if the Operators are not demanding G2S then why should Suppliers spend valuable resource time switching to it. SAS must be good enough.
- IGSA members, who are mostly Suppliers, have a good understanding of G2S capabilities. However, because the Operator member community has diminished greatly, this knowledge is not being shared with them — the consumers.

3.4.8 Operator Conclusion

The following conclusions can be drawn from the operator comments:

Benefit: Using the G2S protocol, “has led to a multi-million dollar savings for our organization.”
- Byron Bridger, AGLC

- Operators confirmed that IGSA standards are providing an extremely compelling value proposition that is far exceeding any of their initial expectations.
- Operators recognize that the awareness IGSA standards provide is invaluable for overall casino operations.
- Not insisting on full IGSA certification by operators was a mistake that has led to unnecessary long-term pains and interoperability issues.
- In the USA the segregation of core business units within the casino operation is working counterproductive to the business objectives of optimizing business revenues. The study demonstrates the power of collaboration between casino operations, casino marketing and IT departments but identifies the large shadow over the industry due to supplier secrecy and control modus operandi.
- Individuals managing gaming floors have more technical knowledge about their notebooks or computers then they do about the multimillion dollar EGM’s and system equipment that is at the core of their business, but they are eager to know.

- The case study further contrasts the significant lack of knowledge between those operators whom have never participated in IGSA and those whom have been an active part of IGSA.
- Operators who implement IGSA standards see value across the board:

Table 3.1 Operator Conclusions

	Without IGSA Standards	With IGSA Standards
Gaming Floor	Serial connections leveraging a vendor proprietary legacy protocol	Open standards extensible protocol, high speed floor
Interoperability	High risk, high cost, high time to market	Certified platforms can be deployed with minimal risk & cost
Responsiveness to Change	Limited	Operator-initiated download and configuration to their EGM from a single central system
Player Engagement	None	Bi-directional communication with players
Business Intelligence	Aggregate data designed to meet the EGM Host's needs	Predictive Analytics and real-time dashboards
Security	Vendor proprietary security solutions	Single deployment of shared components, single view of the player
Gaming Channel Convergence	Monolithic, siloed gaming channels	Open standard extensible protocol, high-speed floor

In conclusion, for operators to protect their gaming investments, they need to mandate devices that support open standards and/or be part of the only organization that is working on furthering innovation and transparency. Industry change can be facilitated to the benefit of the industry and policy domains.

3.4.9 Regulator Input

3.4.9.1 Regulator Vision

Regulators are seeking to remain neutral when it comes to protocols used in gaming. They remain focused on ensuring the integrity of gaming, preventing fraud and protecting players. However, to achieve each of those goals they require data. As gaming has continued to evolve they are finding that newer technologies, including protocols, are more capable of providing them with the tools they need to achieve their objectives. Their vision is to see these newer technologies implemented, and to do so either in partnership with the Industry Domain jointly agreeing to adoption, or by gently nudging the industry in that direction. When necessary, some regulators have no issues with mandating the use of certain technologies.

3.4.10 Regulator Findings: SAS

Note: "SAS has gone as far as it can go. We are starting to see a variety of issues with this protocol."
- Todd Nelson
Missouri
Gaming Commission

Some regulators clearly see that SAS in no longer a viable protocol. Todd Nelson, of the Missouri Gaming Commission, explains:

- "SAS has gone as far as it can go. We are starting to see a variety of issues with this protocol. One example is that suppliers are doing things that the protocol was never designed to do. They are trying to extend its life by doing things such as putting data in buckets they are not supposed to put them into; buckets that were not intended for the purpose they are now being used for. As regulators we are not going to allow that anymore."

IGSA Standards Benefits and Value

Game Authentication Terminal standard (GAT)

This standard is being used in every gaming jurisdiction interviewed.

- In some jurisdictions, Gaming Regulations require that all gaming equipment support the IGSA GAT standard and the regulators rely exclusively on GAT to verify all software, both in the lab and in the field. In other jurisdictions, the regulations specify the methodology that must be met and allow a variety of tools to achieve authentication.
- Prior to implementing GAT there was zero standardization on how gaming components were verified. If we did not have GAT then we would have had to support all of these other verification procedures. We would have had to maintain potentially different verification procedures for every single cabinet. GAT has made it a lot easier to verify the gaming devices.
- GAT and the application we use allow us to collect all data elements needed to store machine specific information. If an issue occurs with a particular piece of software, we can quickly identify where all the machines are and take steps to mitigate the problem. We love the protocol and it is working very well for us.
- Our end goal, the Holy Grail, is to use G2S's GAT capability across the network. That way we can authenticate any

applicable device in the field using a single terminal remotely.

- We have communicated to the suppliers that our goal is to use G2S's GAT so we can verify software over the network.
- We are very interested in using GAT 4.0 which updates the encryption algorithm to SHA3 and includes support for Peripheral Devices.

Certification Database Interface standard (CDI)

This standard was identified as a need by the regulators that participate in the IGSA Regulator's Committee. CDI is being adopted and is seen as starting to provide the expected value.

- We require independent test labs (ITLs) to also support the CDI.
- We don't yet require the suppliers to support CDI, however that is planned.
- The CDI standard is definitely helping us by providing product testing result data in a consistent way from multiple ITLs.

Game to System standard (G2S)

Note: Some regulators are partnering with the industry moving slowly toward full G2S adoption.

This standard is seen as having the most potential to help regulators achieve their objectives more efficiently. However, there are implementation and adoption issues that the regulators are seeking to overcome in a manner consistent with their operational methodologies.

- Some regulators are partnering with the industry moving slowly toward full G2S adoption:
  - As a regulator we were struggling on how to move the industry into the semi-modern era. We realized that things would not change without a nudge, so we are now considering a policy giving operators a reasonable period of time to migrate all their EGMs to be G2S compliant and that those EGMs have multiple port support and that those ports are fully open and accessible by multiple systems.
  - We are considering requiring EGM's to support both G2S and SAS. We have frequent conversations with the suppliers to evaluate their ability to support this functionality as it will allow a regulatory server to talk directly to the EGMs.
  - We were also interested in knowing whether or not they supported simultaneous G2S and SAS communication on their devices. At least at that point we could require

the game to support G2S without disrupting the casino operator's ability to use their legacy accounting system. Enabling us to report on gaming devices by running GAT over the network would be a win for all of us regulators.

Note: Other regulators are taking a more forceful approach seeking to utilize G2S capabilities as soon as possible.

- Other regulators are taking a more forceful approach seeking to utilize G2S capabilities as soon as possible:
  - We are likely to mandate the use of the G2S standard in the same way that we required GAT within our jurisdiction.
  - The large suppliers have a solid support base for G2S but some of the smaller suppliers do not. The policy change would provide those smaller suppliers the incentive and time to catch up with the larger ones.

Regulatory Reporting Interface standard (RRI)

There is strong interest in this standard and how it provides for a single data exposing methodology for both land-based and on-line gaming, including sports betting:

- We believe that there is value in having one standard — IGSA's Regulatory Reporting Interface standard — that provides information from every gaming system/vertical from land-based to online to sports wagering. Expecially if it is aligned to G2S and can provide GAT related data, as an example.
- We see having a single data feed as very positive.

Third-Party Game Interface standard (TPI)

Some regulators are very interested in the value that this standard is providing:

- The integrations between Remote Gaming Systems (RGS) and Internet Gaming Platforms (iGP) was problematic for us. Many Operators are not getting the content they want because of the integration costs for the smaller suppliers. We wish TPI had been out and we could have adopted it before we launched on-line gaming.

3.4.10.1 Regulatory Observations

- Regulators would like to see IGSA take a more active role in creating applications and tools:
  - Provide software tools that are not created by ITL's to support regulators so that they own their own data.

  - Create an application to use GAT that could work with all the suppliers' EGMs (or at least those of the IGSA members) instead of forcing regulators to create their own.
  - Create a GAT application that comes with a database to store the tested and approved software signatures.
- Regulators see the lack of awareness within the industry as a major impediment to implementation of IGSA standards.
  - Many within the industry are simply not aware of what is available and the value that can be added.
  - There is a complete disconnect within the supplier industry pertaining to IGSA standards.
  - Some suppliers have no idea about RRI or CDI. CDI is a standard suppliers can tremendously benefit from. The people responsible for lab submissions have no knowledge of this capability. This is a big deal.

3.4.11 Regulator Conclusion

The following conclusions can be drawn from the regulator comments.

- IGSA is creating standards that are adding value to regulators. Regulators wish that these standards would be more broadly adopted as older technologies are no longer viable.
- Some regulators are realizing that absent a regulatory mandate SAS will continue to exist and perhaps even to proliferate, forcing all to use antiquated technology and holding the industry back. Others are not yet willing to take that approach.
- Some regulators understand that they have the authority, and are even obligated, to require certain technologies, functionality, or processes through mandate. By their very nature, regulations require, or mandate, what suppliers and operators must do to do business legally within this industry.
- Some regulators are looking to implement an intermediate step requiring that gaming devices support both SAS and G2S simultaneously. This will allow regulators and operators to benefit from the additional functionality and data reporting capabilities without impacting the legacy slot accounting system.
- Some regulators have, or are investigating, allowing access to that data over wireless communication eliminating the

need for networked casino floors. This access would require the appropriate technical fire-walls and security measures.

- As regulators see great value in participating in the Regulatory Committee but question why other regulators, including Tribal regulators, are not participating?

3.4.11.1 IGSA Staff Derived Findings

Industry Protectionism and Impediments

- Suppliers believe they know best, but:
  - Supplier's Sales Staff are not trained on protocols, but rather on the game themes, game mechanics, target market and revenue generation (this game is doing 2x the house average down the street...). Consequently, they cannot advise on functionality that is critical to IT and Marketing.
  - Even supplier's that have a G2S-based system do not expose all the data that G2S can collect from an EGM. Instead they decide what data an operator really needs. As a result, they inadvertently 'dumb down' the capabilities inherent to G2S.
  - EGM suppliers are not supporting or enabling the multi-host connectivity that is a core G2S functionality. This prevents operators from connecting multiple systems to the EGMs and gaining access to the machine data independent of the Host supplier's Casino Management System (CMS).
- Slot operations often work within a silo
  - Slot operations is taked with maximizing slot revenue by making the right purchasing decisions within a tight budget. The protocol the EGM speaks is the furthest thing from their mind, as a protocol is not thought to impact revenue.
- Operators in general don't know what they don't know
  - Operators have a very limited understanding of the capabilities that G2S enables because they have not been participating in IGSA and rely solely on their Supplier's Sales staff for information.
  - Operators are uninformed about the benefits of G2S and they are led by equally uninformed Sales staff to purchase EGMs using antiquated protocols with limited technical and data sharing capabilities.

  - Operators are unaware that G2S is a multi-host protocol allowing multiple systems to connect to the same EGM and subscribe to data based on each system's function. The freedom G2S affords Operators to select the best-in-class systems to perform a task is completely lost. They are forced to buy a 'bundled' product from a single supplier.

- Regulator Challenge
  - The argument that Regulators cannot mandate requirements that will cause suppliers or operators to incur costs is a baseless one. Regulators today mandate that suppliers must have their gaming products tested by Independent Test Labs costing those suppliers millions of dollars annually. Obtaining a copy of all the land-based IGSA standards, forunlimited use world-wide, costs a gaming company just $11,200 annually.
  - If regulators do not mandate the use of IGSA standards that can benefit them, then the industry will continue to rely on outdated technology for as long as they can. The larger suppliers have implemented IGSA because it is required in many parts of the world, but they continue to sell older technology in the US because operators do not know better. The smaller suppliers are potentially never going to implement it because they need a reason to do so.
  - The result of the lack of regulatory mandate is that the Gaming Industry in the US is stagnating technologically, while in Europe things are moving forward fast because Regulators are willing to work together and to mandate change. IGSA Europe is currently working with Regulatory Authorities from 13 countries collaborating to create a single pan-European standard.
  - Some regulators in the US are still applying outdated restrictions to gaming devices such as prohibiting them from being accessed via network outside of the casino. This while simultaneously some jurisdictions are allowing internet-based online gaming and many others are rushing to allow internet-based sports wagering. The idea that online wagering is secure but connecting casino-floor games to online networks is not, seems incongruous.

Mandate to Change the Industry

The only way change will happen in the Gaming Industry is when an entity that has power — a regulator — mandates that all EGMs support **both SAS and G2S** and provides a reasonable migration period to enable that change to happen.

It’s the same as saying that by 2030 all vehicles in the US must meet fuel efficiency standards of 54 miles per gallon. The regulator, the enforcing agency, will have to insist on this sort of change, otherwise — left to its own devices — this industry will trod along with the old stuff forever.

What would happen if regulators mandate that every EGM by 2020 support both SAS and G2S?

- This is a small ask for the major EGM suppliers since in addition to SAS, they already have or are implementing G2S to be able to sell slot machines into the growing number of judications that already require G2S.
- This will provide an incentive for smaller suppliers to implement G2S and in the interest of fairness, provide them the time to achieve that.
- Regulators can then install G2S-based reporting system using all the benefits of G2S such as performing remote GAT spot-checks, verifying all software and subscribing to key data in real-time. This can be accomplished without the need for networked floors, by using wireless and appropriate firewalling of local in-casino servers and the outside world.
- Operators are not negatively impacted because they don’t have to change their slot accounting system. On the contrary, Operators are positively impactec because they too can connect to each EGM using systems that co-exist with other systems, to subscribe to all the data G2S provides which older protocols do not.
- System suppliers are also not negatively impacted. They don’t have to develop a G2S-based casino management or slot accounting system.

Who wins?

- Regulators win because they can implement systems that create efficiencies and even help eliminate risks for the operators.
- Operators win because they can tap into all the data that they don’t have access to today.
- Suppliers win because it’s a minor change to support both SAS and G2S on the EGM side and requires no change to Casino Management Systems.
- The industry wins, by becoming more efficient, by harmonizing processes between land-based and on-line and

leveraging the latest technology to further secure and make gaming transparent which directly translates into integrity.

3.5 Bridging the G2S Gap

3.5.1 A White Paper on Game to System (G2S) Viability in the Distributed Gaming Market

(originally printed in October, 2009)

3.5.1.1 Introduction

Many gaming manufacturers refer to "open protocols" to suggest an industry shift that will allow gaming operators to gain more benefits from their products. They paint a picture of a future where products from all manufacturers will work together seamlessly, where operators' game libraries will grow exponentially, and where operators will control gaming content on a whim.

But not all gaming operators are alike. Only a few manufacturers fully understand the complex distinction between commercial casino and government-sponsored, distributed gaming. Unlike commercial casino operations that usually function under one roof, Electronic Gaming Machines (EGMs) in distributed gaming markets are physically scattered throughout a regulated jurisdiction, often with just a few machines per venue among thousands of venues.

The purpose of this document is threefold. First, it aims to clearly define the universal benefits and functions of open standards for all gaming operators by looking at the development of the G2S protocol.

Second, it outlines why manufacturers need to expand the current G2S protocol in order to meet the distributed gaming market's unique needs, and what distributed operators can do to support those efforts.

Finally, it offers a checklist on how to prepare for full adoption of G2S protocols in your distributed gaming operation.

3.5.1.2 The Effect of Proprietary Protocols on Operators

To run efficient, successful gaming operations, jurisdictions need variety. That's why most operators offer their players a selection of EGMs and game content from multiple manufacturers.

Benefit: “The cost savings from standardizing protocols and eliminating inefficiencies caused by incompatible systems is significant, and could be channeled to customer-focused benefits.” - James, R. Maida, CEO, Gaming Laboratories International, in Public Gaming International, March 2009

Central system providers have adapted their products to gaming operators' diverse EGMs by creating languages, or protocols, that allow EGMs to communicate with their systems.

Whether or not a manufacturer is willing to freely share its proprietary protocol information, all central system providers have complete control over any changes to their protocol — for instance, to upgrade functionality. Operators would prefer to have more input and control over those changes.

Some operators have also claimed that the ambiguity of various proprietary protocols can complicate or delay the installation of products in their gaming venues.

Meanwhile, EGM manufacturers spend considerable research and development resources on the adaptation of games to proprietary protocols — instead of developing new and innovative player-focused products.

Game manufacturers prefer an open protocol because it reduces the cost, effort, and delivery timelines of game development, which would otherwise be spent supporting multiple protocols.

Open protocols will also reduce the deployment time for popular games. Vendors will no longer need to test a game in one protocol, and then decide whether or not to port the game to one or more additional protocols.

A single open protocol will also reduce game certification costs by eliminating the need for additional verification for multiple protocols.

Those combined issues have become the impetus behind the open standards movement in the gaming industry.

3.5.1.3 The Ideal Scenario

Since a manufacturer can claim a protocol is "open" while it is still proprietary, the key word in "open standards" is "standard." It refers to a protocol that functions universally among all manufacturers, and is overseen by an independent body that maintains the standard and its evolution.

In a truly open standard, the protocol might remain one entity's intellectual property — but there's no cost to share the technical documentation, and the protocol must conform to a governing body's requirements.

Ideally, open standards provide operators the freedom to take the best products from each manufacturer and integrate them into their gaming operations seamlessly. Operators then have unfettered control over changes and development of their network.

The open standards movement seeks to fundamentally change gaming operators' focus. Operators want to reduce the complexity of hardware and software integration by opening communications between all EGMs, central systems, and other internal systems.

In doing this, the operators and manufacturers can devote more resources to offering diverse and top-performing game content and player services, with the ability to provide and change that content on demand.

3.5.1.4 The International Gaming Standards Association's Role

To make open standards truly open, manufacturers have had to cooperate at unprecedented levels.

In May of 1998, the Gaming Manufacturers Association (GAMMA) was formed with 20 member companies as a non-profit organization to eliminate the technological communication problems that gaming manufacturers and suppliers were facing.

In 2001, the association decided to change its formal name to the Gaming Standards Association, (GSA, www.gamingstandards.com) in order to reflect the growing number of casino operator members.

In 2020, the association changed its formal name to the International Gaming Standards Association, (IGSA, www.igsa.org) in order to reflect the growing number of members in the international arena.

Now, the IGSA continues to grow in scope as well as numbers. More recently, a significant area of association membership growth has been in the lottery sector. Of the 16 members who are operators, eight represent lotteries.

3.5.1.5 Why G2S isn't in every casino. . . yet

The Game to System or G2S protocol is an XML-based open communication protocol that allows networked gaming features like remote configuration and software downloads. The protocol was designed to be extensible so it can be easily upgraded and updated to meet evolving gaming operation requirements.

Today, most casinos connect their EGMs to their central systems via serial protocols — most often Slot Accounting System (SAS).

Serial protocols have evolved over the years to support functionality like ticket validation, Ticket-In, Ticket-Out (TITO), and cashless gaming.

IGSA and the gaming industry recognize that the future of gaming rests in high-speed IP connections, Ethernet networks, and XML data transfer. However, switching to Ethernet-based EGMs could be a costly alternative because it may require upgrading site wiring, upgrading or replacing EGMs to support Ethernet, and purchasing games written for the G2S protocol.

Despite these hurdles, many operators are learning that G2S can be phased in to co-exist with existing slot floor networks.

3.5.1.6 The challenge for distributed gaming operators

G2S isn't a foreign concept to distributed gaming operators. Out of necessity, they've been early adopters of downloadable game technologies. Remote and centralized operation is a fact of life for successful distributed gaming networks. Distributed markets are more than ready to embrace G2S technology because it's simply another step using an already familiar approach.

However, since G2S was initially driven by casino manufacturers, suppliers, and operators, the standard didn't take into account the fundamental differences between commercial venues and distributed government-sponsored markets.

In a casino, local servers enable data collection, and in many cases a local central system assists in the management of the casino floor. All this typically happens under one roof.

But in distributed gaming environments, EGMs scattered throughout a geographic area are controlled through a remotely-located central system. The system often communicates through the most cost-efficient communication option available. That's often a basic dial-up connection using a local site controller based in distributed retailer locations.

Under these constraints, an operator with a low-bandwidth network can use the G2S protocol by "tuning" the protocol. According to IGSA Technical Director Marc McDermott, the operator can choose to conserve bandwidth by making fewer requests for information, subscribing to fewer events, and asking for less associated information from the system.

“The protocols themselves are capable of successfully conveying information” in either high or low-bandwidth environments, McDermott writes in the January 2009 issue of Slot Manager.

“The key for low-bandwidth operations, then, lies in exercising the restraint necessary to request only the information needed instead of the information desired. However, because of the incredible array of information that the IGSA protocols are capable of providing, staying with what is 'needed' may not be an easy task.”

On the other hand, the demand for more information is outpacing the ability to deliver it affordably through existing communication solutions. Many distributed operators are just as interested in knowing how their games are performing during normal and peak operating hours as their counterparts in casinos.

Distributed operators' hunger for information comes from a desire to collect game play data at a more granular level. In the past, a daily summary of cash won and cash played per terminal might have been enough. Today, though, operators are asking for this information at the game level and at intervals through the day.

In order to report on the data, you need to collect it. In order to collect it, you need a communications network to support it.

Therefore, in order for operators to gather data in real-time, they need a more robust communications network.

But investing in a high-speed network upgrade may be too expensive for most distributed operators. Unlike a casino, where a high-speed network can easily be deployed and maintained, distributed operators need to consider the upgrade costs to support as many as thousands of sites across many different communication networks. While high-speed networks are more readily available, and costs are coming down, some locations still don't have access to high-speed networks. Their alternatives include communications networks like radio, dial-up, and satellite. In some cases, distributed gaming operators have had to consider multiple types of communication to support different geographic locations.

Operators also need to assess whether or not their legacy terminals support G2S requirements, including Ethernet communications and the performance of an XML-based protocol. If not, they would also need to upgrade to more powerful EGMs. In the worst-case scenario, they would need to purchase new EGMs that support those requirements.

While distributed operators have to consider how they can support existing G2S requirements, their operations could also require specific functions not available with the current G2S protocol, such as:

Benefit: GTECH® developed an extension of EGM operating hours to ensure that distributed market operators can meet jurisdictional legislation governing hours of operation.

- Setting Video Lottery Terminal (VLT) operating hours
- Allowing remote VLT master resets without the need for an on-site technician
- Allowing the system to initiate a cashout on the VLT, prior to maintenance or as part of a responsible gaming feature
- Configuring VLT cash limits, which may be dictated by jurisdictional regulations
- Optimizing download time and download messaging/ enabling
- Ability to enable/disable and specify time for Gaming Authentication Terminal (GAT) verification
- Communicate VLT storage space capacity to the central system
- The ability to use a transport mechanism designed for slower communication models.

Benefit: The extension ensures that EGMs store operating hours locally so they can self-disable during non-operational hours. Over the next year, GTECH will be submitting even more extensions in order to make the G2S protocol fully supportive of distributed market operations' needs.

3.5.1.7 The solution for distributed operators

How can the industry adapt the full capabilities of the G2S protocol to the realities and demands of distributed environments?

Through the creation of extensions.

The inclusion of extensions to the protocol must be done in strict accordance with IGSA policy.

An extension is a new class or part of a protocol that a manufacturer develops to extend, or ad functionality to, a IGSA protocol. The IGSA typically reviews a manufacturer's extension to ensure compatibility, and if accepted, approves the extension by a formal vote by one of IGSA's technical committees. Once approved, the extension becomes part of the respective standard applied by all IGSA-compliant operators, and is maintained by the IGSA.

However, most manufacturers haven't been compelled to invest the effort into making the changes needed to extend the functionality of the G2S protocol for the distributed space.

That's why the manufacturers who understand the specific needs of the distributed gaming market are best suited to provide those solutions.

In June 2009, the IGSA approved the first-ever manufacturer-submitted extensions designed specifically for the distributed gaming market. As more extensions are developed for the distributed market, distributed operators will soon be able to benefit from G2S.

Operators also play a key role in supporting those efforts. By becoming an active member of IGSA and its technical committees, your support can help advance the cause for distributed operators and ensure that distributed-market extensions are accepted as part of the G2S open standard.

3.5.1.8 Making G2S work in distributed markets

There are strategies you can put in place right now to prepare your distributed operation for open standards.

- Ensure that your system can capitalize on G2S protocol-supported functions, or at a minimum, be able to support G2S as it gains acceptance and popularity.
- Ensure that your system can collect more data from the terminals in the distributed market, and use that data to optimize your terminal and gaming network, including performance, diagnostic and security reporting.
- Ensure that the systems and EGMs that you install are G2S compliant — in other words, that they conform to IGSA standards and are poised for interoperability.
- Your EGMs should support being fully downloadable and network gaming ready. They should feature upgradeable memory capability, extendable storage space for games, and secure data links.
- The system should be field tested and proven in the distributed market. This is a critical element.
- The system should support the additional security needed to monitor and control terminals in a distributed market that are not under the watchful eyes of staff dedicated to walking the floor and monitoring security cameras — protection that's typically available in casinos, but difficult to implement in the distributed environment.
- The system needs to be able to enable or disable gaming machines across a distributed network and not rely on site operators to disable the games outside of regulatory hours.
- To avoid substantial startup costs, the system should optimally support your existing EGM network and provide a way of supporting terminals as they are upgraded or replaced. This would include the simultaneous support of legacy protocols and newer protocols like G2S.
- The system should allow games to be downloaded, installed, scheduled and switched on an EGM. This will potentially save millions of dollars in operational expenses

needed to add new games to a large network, and allow the distribution of new games in a fraction of time.

- The system and EGMs you purchase should also come with experienced, knowledgeable, and dedicated technical support.

3.5.1.9 The future is open

Benefit: "As a buyer, you will now be more firmly positioned in the driver's seat and can make decisions that are based on the right product mix, unique product features, quality of service, preference, compatibility, openness, data access, and flexibility." - Peter DeRaedt, President, Gaming Standards, Association, In Public Gaming International, Nov 2008

By equipping your operations with end-to-end, G2S-compliant solutions, you'll be ready to embrace G2S when the extensions you need soon become available. It will open up new functionalities, make your operation more nimble and versatile, and create efficiencies and significant cost savings in multiple areas of your operation.

You'll be able to choose a IGSA-compliant vendor without thinking twice about product compatibility. You'll be able to cement players' loyalty by offering the games they want exactly when they want them. You'll have even more ways to implement responsible gaming features on your EGMs.

In short, you'll be securing your future as a successful distributed gaming market operator.

3.6 Case Study for the Value of IGSA Protocol Implementation

This case study presents the level of computer interoperability that existed in Canadian gaming jurisdictions prior to integrating IGSA protocols in their operations, the level of interoperability that currently exists, and the goals for future interoperability using IGSA protocols.

(originally printed in June, 2013 as a PowerPoint presentation. Paraphrased for improved text flow)

3.6.1 Gaming Technology Modernization

A long term transformational initiative intended to evolve a gaming Service-oriented architecture (SOA) through system enhancement, acquisition, integration and development.

3.6.1.1 Technology Priorities

The technology priorities identified as goals achieved by implementing IGSA standards are threefold.

First, the technology must enable business needs across gaming streams through improved product analytics and gaming channel convergence.

Second, the technology must improve operational effectiveness by increasing business agility and reducing time to market.

Third, the technology must understand players and actively manage player relationships providing a single view of the player across all gaming channels. Additionally, the player experience must be enhanced.

3.6.1.2 Legacy Environment

The Canadian gaming jurisdictions reviewed in this case study operate large gaming enterprises across multiple channels. This creates a complex operating model due to many factors, – such as:

- different retail operating agreements/models
- Best of Breed solutions that are fit for purpose to the gaming channel are inherently multi-vendor
- siloed transactional data is created in each gaming channel

Business environment and consumer expectations of play experience require a more sophisticated approach to integration.

Relying on older, legacy protocols created technical barriers. Because of the limitations of SAS, any type of solution to a communications issue was tied to vendor proprietary protocols; a tremendous revenue generator for the vendor.

Note: SAS, the casino floor protocol of the past, is no longer supported.

Integration capabilities of any kind were limited and complex. There was a lengthy time to market for new features and games. Lastly, using SAS was, in effect, keeping the technology on your gaming floor in an antiquated state, as the SAS protocol is no longer supported.

3.6.1.3 Architectural Requirements

Architectural requirements as a result of integrating IGSA protocols were many. Canadian gaming jurisdictions wanted to maintain ownership and control over Enterprise integration decisions as the Solution Integrator. They also wanted to maintain ownership and control over key data.

Additionally, they wanted the ability to analyze and make business decisions from an Enterprise perspective, improve time to market and operational processes, minimize the effects of Vendor Lock-in, and minimize the effects of Operator Lock-in. Lastly, the Canadian gaming jurisdictions wanted to ensure a single view of the player.

3.6.1.4 Current Environment

The first segment of Canadian gaming to convert to IGSA open protocols was VLT (Video Lottery Terminal). Once the conversion had taken place, the benefits derived immediately were many.

Previously, rich, detailed data was not available to be captured for analysis.

- The data available using IGSA protocols is not dependent on the needs of the host system.

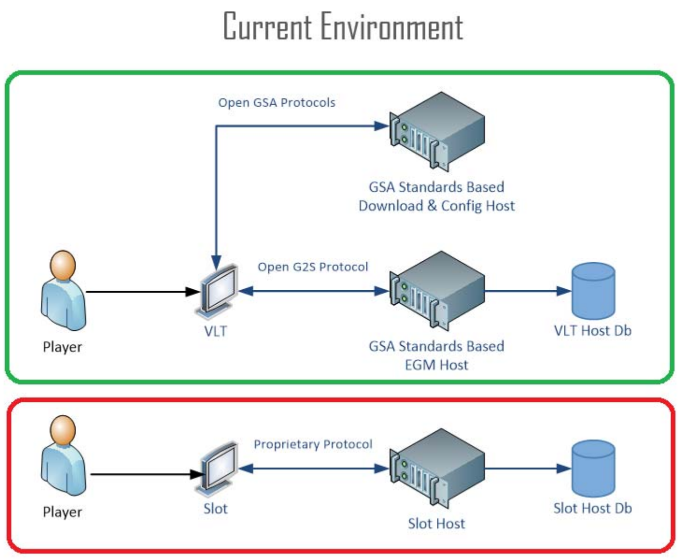
Previously, remote download and configuration from a central platform was a much desired, but impossible to implement feature.

- With G2S, the operator can change game sets, operating systems, and peripherals remotely.

With G2S, you have a single, industry standard secured implementation. Additionally, with G2S there is a significantly improved time to market for new EGMs due to simplified integration and the ability to deploy an increased variety of platforms and products.

Lastly, with G2S functionality can easily evolve to meet business and player needs because IGSA protocols are extensible.

However, note that casino and lottery markets are still using proprietary protocols. The illustration below outlines how a typical network topology in a Canadian gaming jurisdiction looks using IGSA standards (green area), and using a proprietary protocol (red area).



3.6.1.5 Building on the Foundation

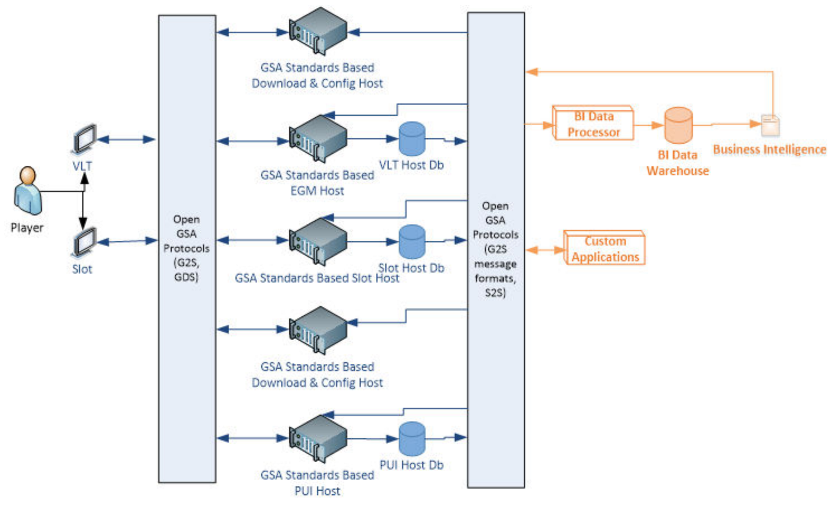
With a goal of delivering the right game in the right place at the right time, IGSA standards help turn this idea into reality.

With easy access to desired, robust data, and easy remote download and configuration, this provides the operation with increased agility to respond to changes and market demands. Integrating systems facilitates both a single player view and convergence of gaming channels. This helps the operation be well positioned for future market conditions.

Improved product analytics across gaming streams provides many benefits. You'll have a more focused ability to make informed product purchasing and placement decision. Enjoy a better understanding of the financial performance of your casino floor network. Take advantage of the ability to make comparisons across gaming channels.

Having a single player view provides many benefits. From an operations perspective, you will have a comprehensive view of player behavior across channels. Also, your players will enjoy a seamless and consistent personal customer experience. They will benefit from a consistent implementation of Social Responsibility programs. You'll now have the ability to cross promote and deliver product to the player where and when they wish to play it.

A network topology with IGSA protocols fully integrated into the casino operation will look something like the illustration below.


GSA Standards Based Download & Config Host
VLT
Player
Slot
Open GSA Protocols (G2S, GDS)
GSA Standards Based EGM Host
VLT Host Db
GSA Standards Based Slot Host
Slot Host Db
GSA Standards Based Download & Config Host
GSA Standards Based PUI Host
PUI Host Db
Open GSA Protocols (G2S message formats, S2S)
BI Data Processor
BI Data Warehouse
Business Intelligence
Custom Applications

3.6.2 Business Intelligence Case Study

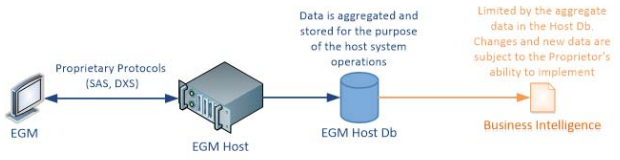
3.6.2.1 Where we were

Table 3.2 The Canadian Experience - Where we were

	Where we were
Gaming Floor	Serial connections leveraging a vendor proprietary legacy protocol
Interoperability	High risk, high cost, high time to market
Responsiveness to Change	Limited
Player Engagement	None
Business Intelli-gence	Aggregate data designed to meet the EGM Host's needs
Security	Vendor proprietary security solution
Gaming Channel Convergence	Monolithic, siloed gaming channels

3.6.2.2 Previous State — Business Intelligence

The illustration below displays how proprietary protocols (such as SAS) limit using the data for anything other than host system operations. Any changes to the data and any new data are subject to the proprietor's ability to implement it.


Proprietary Protocols
(SAS, DXS)
EGM
EGM Host
Data is aggregated and stored for the purpose of the host system operations
EGM Host Db
Limited by the aggregate data in the Host Db. Changes and new data are subject to the Proprietor's ability to implement
Business Intelligence

3.6.2.3 Business Intelligence Challenges

Without IGSA protocols, the challenges inherent for an operator when trying to effectively use the data for a customer's benefit are many.

- Data is aggregated and filtered based on the System Provider's needs.
- Operator's ability to respond to change is dependent on the System Provider's ability to deliver data.
- Each gaming channel may have data in different formats or aggregates.
- Significant effort is needed to attempt to create a single view of the Enterprise.

Without a common communication layer straddling all logical devices in use on the casino floor, the casino operator faces enormous challenges in providing players with a satisfying user experience.

3.6.2.4 Where we are

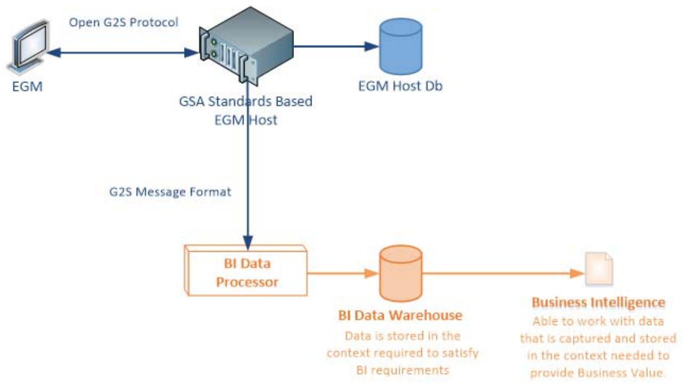
Table 3.3 The Canadian Experience - Where we are

	Where we are
Gaming Floor	Migrating to an open standard protocol, high speed enabled floor
Interoperability	Decreased risk & cost and faster time to market. Increased ability to deploy a variety of platforms.
Responsiveness to Change	Download and configuration to EGMs from a single central system
Player Engagement	Developing standards for PUI
Business Intelligence	Collecting detailed play data
Security	Industry standard security model

Table 3.3 The Canadian Experience - Where we are

	Where we are
Gaming Channel Convergence	Groundwork in place for convergence

3.6.2.5 Current State — Business Intelligence


Open G2S Protocol
EGM
GSA Standards Based EGM Host
EGM Host Db
G2S Message Format
BI Data Processor
BI Data Warehouse
Data is stored in the context required to satisfy BI requirements
Business Intelligence
Able to work with data that is captured and stored in the context needed to provide Business Value.

3.6.2.6 Business Intelligence — Immediate Benefits

Integrating G2S into a casino operation provides immediate improvements. First, G2S provides easier access to rich and unfiltered information. This volume of rich information can paint a very compelling picture of the true activity on the gaming floor, such as player behavior and machine activity.

Another significant advantage to having G2S operating on your floor is that any changes to BI requirements are simple to implement and do not require a complete regression test of the regulated EGM host system. Lastly, the BI data processor is portable between any G2S enabled gaming channel.

3.6.2.7 Where we are going

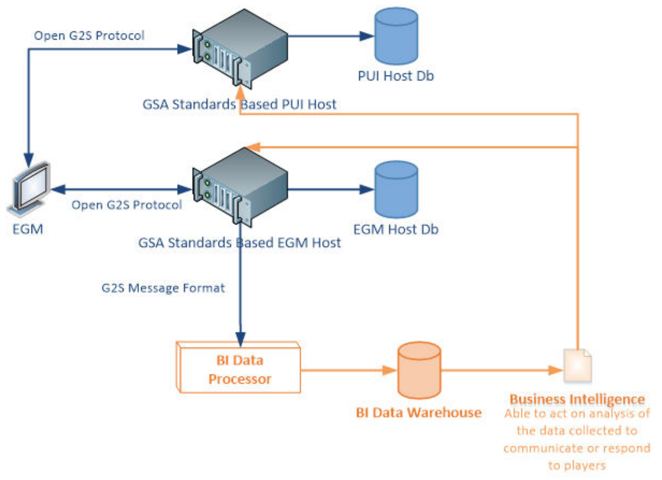
Table 3.4 The Canadian Experience - Where we are going

	Where we are going
Gaming Floor	Open standard extensible protocol, high speed floor
Interoperability	Certified platforms can be deployed with minimal risk and cost

Table 3.4 The Canadian Experience - Where we are going

	Where we are going
Responsiveness to Change	Player initiated download and configuration to their EGM from the single central system
Player Engagement	Bi-directional communication with players
Business Intelligence	Predictive analytics, real-time dashboards
Security	Single deployment of industry standard security model across channels
Gaming Channel Convergence	Single deployment of shared components, single view of the player.

3.6.2.8 Future State — Business Intelligence


Open G2S Protocol
PUI Host Db
GSA Standards Based PUI Host
EGM
Open G2S Protocol
EGM Host Db
GSA Standards Based EGM Host
G2S Message Format
BI Data Processor
BI Data Warehouse
Business Intelligence
Able to act on analysis of the data collected to communicate or respond to players

3.6.2.9 Business Intelligence — Future Benefits

Convergence of the gaming channels will provide a single view of the player and an Enterprise view of key data. Operators will have a better understanding of the player and can be more responsive to player needs. Rich, detailed data will enable Operators to deliver personalized messaging to players and make informed purchasing and placement decisions.

Lastly, an Enterprise view of key data, inclusive of unstructured data, will enable Operators to identify trends and analyze how factors, such as weather or a promotion at a single channel, impact the entire network.

Table 3.5

	Where we were	Where we are	Where we are going
Gaming Floor	Serial connections leveraging a vendor proprietary legacy protocol	Migrating to an open standard protocol, high speed enabled floor	Migrating to an open standard protocol, high speed enabled floor
Interoperability	High risk, high cost, high time to market	Decreased risk & cost, and faster time to market. Increased ability to deploy a variety of platforms.	Decreased risk & cost, and faster time to market. Increased ability to deploy a variety of platforms.
Responsiveness to Change	Limited	Download and configuration to EGMs from a single central system	Download and configuration to EGMs from a single central system
Player Engagement	None	Developing standards for PUI	Developing standards for PUI
Business Intelligence	Aggregate data designed to meet the EGM Host's needs	Collecting detailed play data	Collecting detailed play data
Security	Vendor proprietary security solution	Industry standard security model	Industry standard security model
Gaming Channel Convergence	Monolithic, siloed gaming channels	Groundwork in place for convergence	Groundwork in place for convergence

3.7 Case Study - The Inconvenient Truth - of G2S Adoption

3.7.1 Why G2S Has Not Been As Widely Adopted As Expected

As the saying goes, money makes the world go around. It is money, or the potential for losing revenue, that has caused Gaming Suppliers not to willingly and globally adopt G2S. However, before going into does details it is important to first understand why G2S is so much better than SAS.

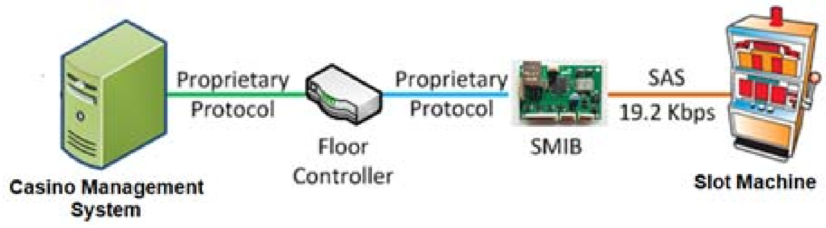
The SAS protocol was developed over three decades ago in the era before the Internet was invented before true networks were invented, and before data security was a real need. This is in sharp contrast with G2S that was developed using technology that is currently widely used in Internet banking and Internet commerce, supports true broadband networks, and has data security through encryption, built in.

The table below provides a quick glance top level comparison between SAS and G2S.

Table 3.6

Description	SAS	G2S
Extensibility - can have functionality added to support innovation without breaking backwards compatibility.	NO	YES
Securely communicates data.	NO	YES
Support collecting and transmitting all the data slot machines generate.	NO	YES
Support high-speed broadband networks using TCP/IP.	NO	YES
Supports having multiple systems connected to and communicating with each slot machine.	LIMITED	YES
Provides a full inventory of software and firmware, including version number, for each slot machine.	NO	YES
Provides a method to automatically and remotely (don't have to be physically on the casino floor) authenticate and verify slot machine critical software.	NO	YES

So, given how much more capability G2S has why has not the world jumped to adopt it. To understand the reasons behind this we must first understand how the SAS slot machine to Casino Management System (CMS) ecosystem works.



The SAS slot floor ecosystem is comprised of (in the illustration above, from right to left) the slot machine, a slot machine interface board (SMIB), an optional floor controller and the Casino Management System (CMS).

The CMS, optional floor controller and SMIB are sold by the system supplier as part of a system solution. The slot machine is sold by gaming manufacturers. Many gaming manufacturers are also system suppliers, but not all are.

There is a common misunderstanding that Casino Management Systems use SAS. This is not correct. The only components within this ecosystem that use SAS are the slot machine and the SMIB. The SMIB is a protocol translator, translating SAS into the proprietary protocol used by the floor controller and CMS.

3.7.1.1 Now Let’s Follow The Money

SAS being an old protocol, has been developed by gaming manufacturers for years. Since it has changed very little over the three decades, there has been very little investment in money needed from these manufacturers. In other words, it is a fixed cost and they know how to develop it quickly, cheaply and effectively.

The system suppliers have been developing SMIBs since when SAS was originally invented. The SMIBs back then acted as the serial communication device because slot machines did not have that capability. Over time system suppliers have added functionality to SMIBs to control user displays, keyboards, card readers, printers, bill validators, etc. The SMIB today is, in addition to a protocol translator, a peripheral device controller. That is why system suppliers charge thousands of dollars per SMIB.

Since the slot machine sends SAS to the SMIB and the SMIB translates SAS to the CMS proprietary protocol, and since SAS can only support one or two systems connected to it, all of the functionality that a casino operator needs to run the casino must be part of, or connect to, the CMS. This means that all innovation is controlled, to a variety of extent, by the CMS supplier.

As a friend of mine Faruk, pointed out, this is like you wanting ESPN. You can only get it from your cable company and must pay whatever they charge. You cannot simply connect another cable company's device to your TV and get only ESPN from them! The CMS suppliers have created the same monopoly.

3.7.2 Why Do System Providers Consider G2S Such a Threat?

Benefit: With G2S, the operator is not tied to the CMS supplier. Any subsystem (marketing, analytics, maintenance, monitoring, signage, and progressives, etc) may be used.

Well, G2S supports having an almost unlimited number of systems connected to each slot machine. This breaks the monopoly that the CMS suppliers have. In the cable company example, G2S allows consumers to get any channel from any supplier. In the casino world, that would allow casino operators to shop the marketplace for solutions such as Marketing Systems, Analytics Systems, Maintenance Systems, Monitoring Systems, Signage Systems, Progressive Systems, etc. and not be forced to buy these solutions from the CMS supplier.

G2S breaks that monopoly that CMS suppliers have today and will, eventually, impact the amount of revenue they will generate because they will be forced to compete with companies that focus on a single solution. And these companies may be more responsive to casino operator needs, and may deliver better functionality, faster and cheaper than the CMS suppliers.

Ironically, CMS suppliers will still be able to make lots of money from the sale of SMIBs if slot machines all used G2S rather than SAS. Instead of the SMIBs translating SAS to the CMS proprietary protocol, the SMIB would translate G2S to the CMS proprietary protocol.

3.7.3 Why Are Game Manufacturers Not Implementing G2S?

Once again this is simple economics. Game manufacturers are in the business of developing slot machine cabinets and slot machine content, better known as games. Casino operators, namely the casino floor management (VP of Operations, VP of Slots, Slot Director or Performance Manager) is responsible for maximizing the revenue generated through those slot machines. They do this by purchasing the most popular and profitable slot machines and content. They do not care about technology! They don't take into consideration SAS versus G2S, data security, data transmission speeds, access to additional data, increased flexibility, or even risk management. They only focus on revenue generation.

Since casino operators do not demand G2S, game manufacturers do not develop it. They are happy to use the same old SAS protocol, which again has changed little over a decade, and focus their energy on developing creative, compelling and competitive content!

3.7.4 How Do We Change This?

The key to change lies with the regulators and casino operators.

The regulators need to understand that SAS has zero data security. To compensate for this, lots of physical things have been added. Locked, metal CPU compartments, notifications of when software is going to change, requiring slot floor isolated networks, not allowing slot machines to connect to the Internet, and a myriad other band-aids!

The regulators also need to understand that they can get much more data transparency relative to the integrity of the software running in slot machines via G2S. In addition, G2S allows them to connect a separate monitoring system to provide better oversight of slot machines.

Casino operators beyond casino floor management need to be engaged in the CMS and slot machine purchasing decisions. The CMS and slot machine are technology and complicated technology at that. Therefore, the Chief Information or Chief Technology Officer needs to be involved.

Benefit: Data analytics normally fall under the control of finance or marketing. Therefore, senior management in these departments need to be involved in the purchasing decisions.

Since today businesses understand that data is the key to growth, and since data analytics is usually within the control of Finance or Marketing, the Chief Financial and / or Chief Marketing Officer needs to be involved. If the CIO / CTO and CFO / CMO are included in the purchasing decisions, they will demand G2S over SAS.

G2S allows the CIO / CTO to utilize all the tools they use to secure and manage their back-of house networks and deploy that to the much more critical casino floor network. The CFO / CMO can gain access to all the data slot machines generate and deploy analytical, marketing, promotional tools that provide them better management and revenue generation capabilities.

Getting these individuals to understand what G2S provides and why it has not been adopted is truly the only way we can achieve broad adoption of this powerful and future-proof technology.

3.7.5 IGSA Standards: Case Study Takeaways

3.7.5.1 Operators

- The level of operational efficiency resulting from implementing this protocol versus the previous way things were done using the older protocol, has led to a multi-million dollar savings for our organization.
- IGSA Certification is the critical requirement to get to interoperable solutions. Our inability to insist on IGSA certification, led to both short-term and long-term pains.
- In the US the segregation of core business units within the casino operation is working counterproductive to the business objectives of optimizing business revenues.

3.7.5.2 Regulators

- SAS has gone as far as it can go. We are starting to see a variety of issues with this protocol.
- We love the GAT protocol and it is working very well for us.
- Some regulators are looking to implement an intermediate step requiring that gaming devices support both SAS and G2S simultaneously.
- Some regulators allow the use of wireless technology with the appropriate IT security, to enable EGM connectivity to multiple hosts without requiring expensive casino-floor network wiring.

3.7.5.3 Suppliers

- Suppliers are not actively selling G2S in the US even though they have developed it and sell EGMs using G2S in Canada and several European countries where G2S is required.
- Supplier sales personnel do not understand the capabilities that G2S provides over SAS. They are not trained on the value that G2S can provide operators, but instead focus on selling EGMs and game themes based on their performance.

CHAPTER

4

Infrastructure Classes

4.1 Introduction

The infrastructure classes are the foundation of G2S. All EGMs will implement the three core infrastructure classes: `communications`, `eventHandler`, and `meters`.

The `auditMeters` class is optional.

These classes keep communications going between the EGM and host systems. They also allow host systems to set event and meter subscriptions so that event and meter information is provided in real-time.

4.2 communications

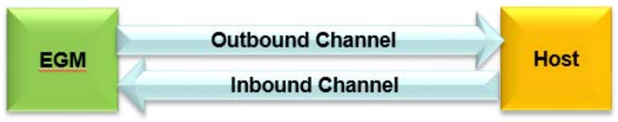
The `communications` class controls the flow of messages between the EGM and host systems. It is also used to discover the set of devices installed in the EGM. The `communications` class is responsible for managing both point-to-point communications and multicast communications.

The `communications` class uses two of the four G2S *Building Blocks*: status and profile. There are no meters or transaction logs associated with the `communications` class.

4.2.1 Communications Associations

Benefit: Direct communications between the EGM and multiple host systems — no SMIBs required.

A communications association is the logical connection between an EGM and a host. Each communications device is used to manage a single communications association. Each communications association requires two communications channels; an inbound channel and an outbound channel.



4.2.2 Getting Connected

Benefit: Network-based communications using 100Mb/sec TCP/IP connections with TLS 1.2 for security.

The EGM always initiates communications, opening the outbound communications channel to the host. EGMs are configured to know the locations of the hosts. Hosts are not configured to know the locations of EGMs.

When the HTTP/SOAP transport is used, the EGM announces its network location to the host when it establishes outbound communications.

The host must use the network location to open the inbound communications channel back to the EGM. Two HTTP/SOAP connections are needed.

When the WebSocket transport is used, the same WebSocket connection is used for both the outbound communications channel and the inbound communications channel. Only one WebSocket connection is needed.

Once the communications association is established, the EGM and the host can start exchanging messages.

4.2.3 Communications States

There are five possible communications states; closed, opening, sync, online, and overflow. The illustration below illustrates the communications states and the transitions that occur when an EGM comes online.

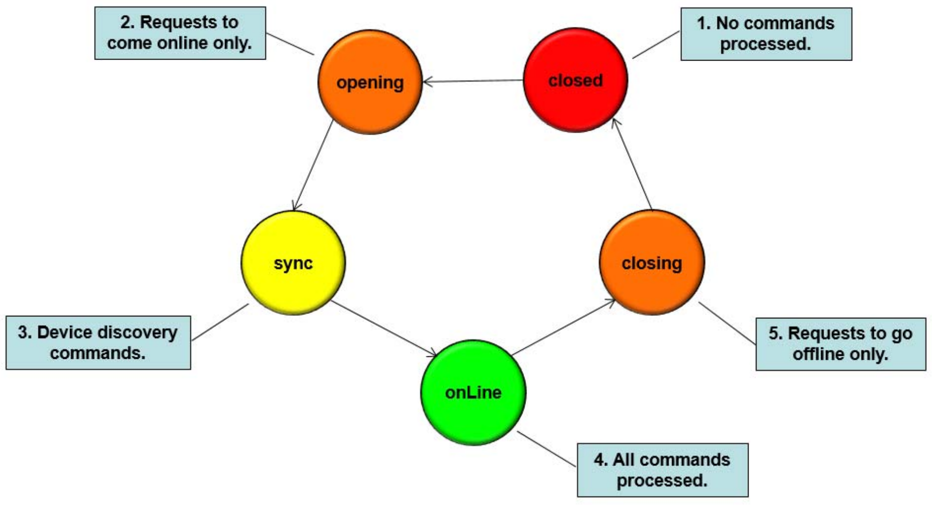
Initially, communications are *closed* (#1). In this state, there is no communications between the EGM and the host.

When ready, the EGM will transition to the *opening* state (#2) and send a request to come online. The EGM will continue to send `commsOnLine` commands until the host sends back an application-level acknowledgement.

After the host acknowledges the `commsOnLine` command, the EGM will transition to the *sync* state (#3). While in this state, the host can interrogate the EGM, find out its capabilities, check configurations, and more.

When the host is finished interrogating the EGM, it will send a `setCommsState` command to the EGM telling the EGM to transition to the *onLine* state (#4). Once in this state, the EGM will commence normal operations, generating commands as necessary.

To end communications, the EGM will transition to the *closing* state (#5) and send a `commsClosing` command to the host, letting the host know that it will be ending communications. If the host responds, communications will be terminated immediately, otherwise the EGM will wait 30 seconds and then terminate communications.


2. Requests to come online only.
opening
closed
1. No commands processed.
sync
closing
3. Device discovery commands.
5. Requests to go offline only.
onLine
4. All commands processed.

4.2.4 Command Sequence

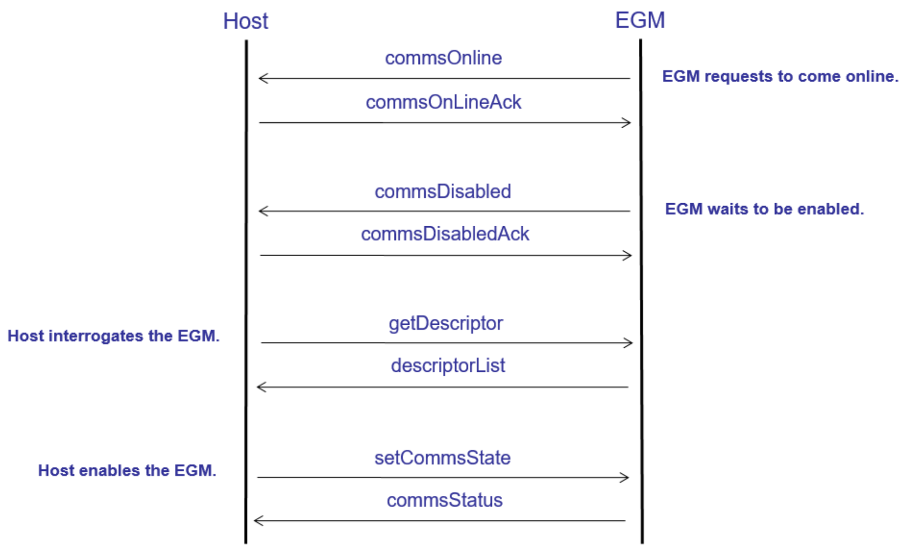
The sequence diagram below illustrates how commands in the `communications` class are intended to be used.

In G2S, communications is always initiated by the EGM with the `commsOnline` command. Host systems are not aware of an EGM until the EGM initiates communication. The EGM will continue to send `commsOnline` commands until the host sends back an application-level acknowledgement.

The EGM then waits for communications to be fully enabled. While waiting, the EGM will only respond to requests from the host; it will not generate any requests of its own, except for the `commsDisabled` command. The EGM will periodically send the `commsDisabled` command to remind the host that communications are disabled.

Benefit: Descriptor lists provide a complete inventory of the logical and physical devices within the EGM including vendor, product, release, and serial number.

During this period, the host can interrogate the EGM with the `getDescriptor` command to find out which G2S devices are installed in the EGM. The EGM responds with the `descriptorList` command. When ready, the host instructs the EGM to start full two-way communications with the `setCommsState` command.



4.2.5 Commands

Table 4.1 `communications` Class Commands Originated by EGM

Command	Description
`commsOnline`	This command is used to establish communications with the host.
`commsDisabled`	This command is used to notify the host that full two-way communications is disabled.
`commsClosing`	This command is used to notify the host that communications is ending.
`getMcastKeyUpdate`	This command is used to request a new set of encryption keys for multicast communications.
`keepAlive`	This command is used to test the communications channels.

Table 4.2 `communications` Class Commands Originated by Host

Command	Description
`getCommsProfile`	This command is used to request the current configuration for a `communications` device.
`setCommsState`	This command is used to enable or disable full two-way communications with an EGM.
`getCommsStatus`	This command is used to request the current status of communications between an EGM and the host.
`getDescriptor`	This command is used to request the list of devices installed on the EGM.
`joinMcast`	This command is used to request that the EGM join a multicast group.
`leaveMcast`	This command is used to request that the EGM leave a multicast group.
`getMcastList`	This command is used by a host to request the list of multicast groups that the EGM has joined.
`mcastKeyUpdate`	This command is used to change the encryption key for a multicast group.

Table 4.2 `communications` Class Commands Originated by Host

Command	Description
`setKeepAlive`	This command is used to set the maximum interval without inbound or outbound communications before a `keepAlive` command is sent from an EGM.
`keepAlive`	This command is used to test the communications channel.

4.2.6 Events

Table 4.3 `communications` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Comms All Faults Cleared
Comms Established
Comms Not Established
Comms Outbound Overflow
Comms Outbound Overflow Cleared
Comms Inbound Overflow
Comms Inbound Overflow Cleared
Join Multicast Group
Leave Multicast Group
Multicast Message Error
Security Parameters Updated
Comms Host Unreachable
Comms Transport Up
Comms Transport Down

Table 4.3 `communications` Class Events

Event
Certificate Reenrollment Failure
Certificate Reenrollment Failure Cleared

4.2.7 Meters

There are no meters associated with the `communications` class.

4.3 eventHandler

The eventHandler class manages the event subscriptions for an EGM. It provides the means for the host to determine the events supported by the EGM and then set a subscription for those events. Based on the subscription, the EGM collects events and delivers them to the host. Each host can have its own eventHandler device.

Event delivery can be best effort, where no response is expected, or guaranteed, meaning the EGM will wait for acknowledgement before discarding the event.

Benefit: The meters, status, and logs affected by an event can be sent with the event.

The host can also request that the data affected by an event be sent with the event.

The eventHandler class uses three of the four G2S *Building Blocks*: status, profile, and logs. There are no meters in the eventHandler class. As opposed to other classes, each eventHandler device has its own log so that the subscriptions of one host do not affect the subscriptions of another.

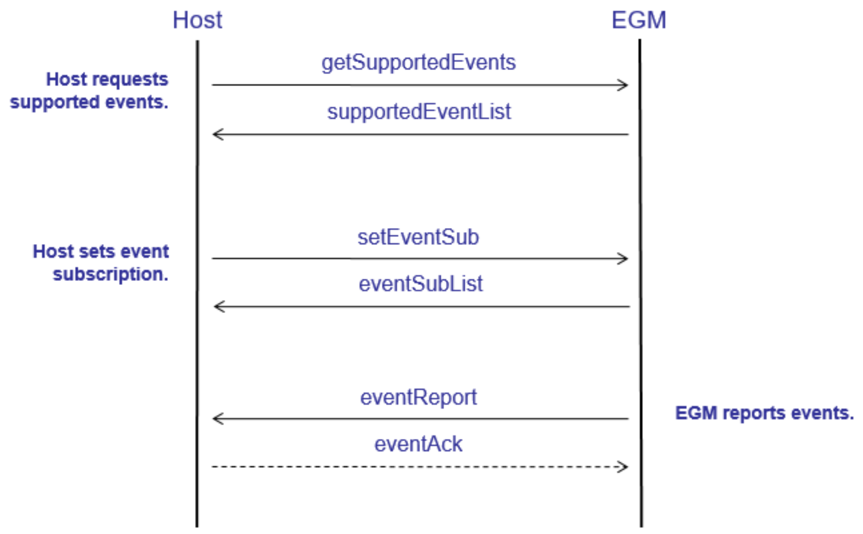
4.3.1 Command Sequence

The sequence diagram below illustrates how commands in the eventHandler class are intended to be used.

First, the host sends a getSupportedEvents command to the EGM to determine which events are supported by the EGM.

After the EGM responds with the supportedEventList command, the host uses the setEventSub command to set its event subscription.

As events occur, the EGM reports the events to the host using the eventReport command. If guaranteed delivery is being used, the host responds with an eventAck command.



4.3.2 Commands

Table 4.4 `eventHandler` Class Command Originated by EGM.

Command	Description
`eventReport`	This command is used to deliver events and affected data to the host.

Table 4.5 `eventHandler` Class Commands Originated by Host

Command	Description
`getEventHandlerProfile`	This command is used to request the current configuration of an `eventHandler` device from the EGM.
`setEventSub`	This command is used to add events to a host's event subscription.
`getEventSub`	This command is used to get a complete list of the events to which a host has subscribed.
`clearEventSub`	This command is used to remove events from the host's event subscription.
`getEventHandlerStatus`	This command is sent to query the status of the `eventHandler` device.

Table 4.5 `eventHandler` Class Commands Originated by Host

Command	Description
`setEventHandlerState`	This command is used to enable or disable an `eventHandler` device.
`getEventHandlerLogStatus`	This command is used to request the current status of the event log for an `eventHandler` device.
`getEventHandlerLog`	This command is used to request the contents of the event log for an `eventHandler` device.
`getSupportedEvents`	This command is used to query the EGM for a list of supported events.

4.3.2.1 Events

Table 4.6 `eventHandler` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Event Subscription Changed
Event Handler Queue Overflow
Event Handler Queue Overflow Cleared

4.3.3 Meters

There are no meters associated with the `eventHandler` class.

4.4 meters

The `meters` class is used to collect meter information from an EGM. Meter information can be requested by a host in real-time or a host can set a meter subscription. When a meter subscription is set, the EGM will send the requested meter information at scheduled times or when certain triggers occur. Each host can be assigned a `meters` device and, thus, set its own meter subscriptions. There are no restrictions on which meters a host can access.

The `meters` class only uses one of the four G2S *Building Blocks*: meters. There are no statuses, profiles, or logs associated with the `meters` class.

4.4.1 Command Sequence

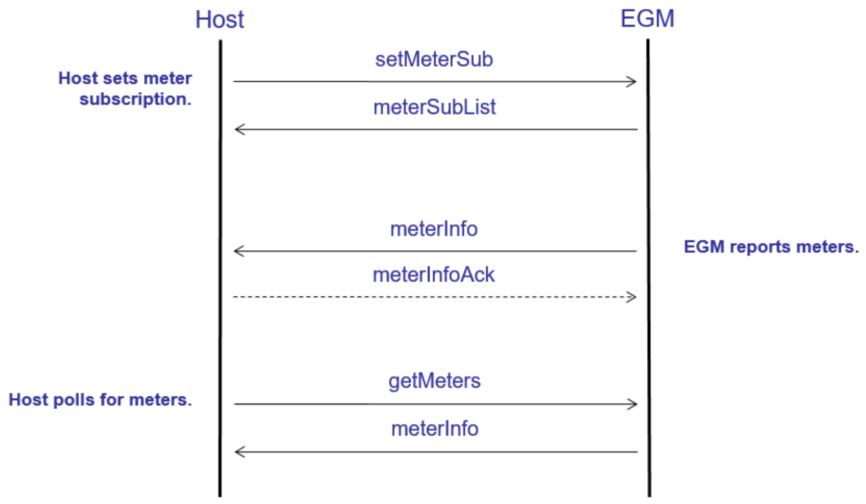
The sequence diagram below illustrates how commands in the `meters` class are intended to be used.

First, the hosts uses the `setMeterSub` command to set its meter subscription.

Benefit: Meters are always in balance when reported.

Next, based on the subscription, the EGM generates `meterInfo` commands containing the meters and sends them to the host. All meters for the subscribed classes and devices are sent. The host does not pick individual meters. The EGM simply sends them all.

Optionally, the host can use the `getMeters` command to poll for meters. A `meterInfo` command is sent as the response.



4.4.2 Commands

Table 4.7 `meters` Class Commands Originated by EGM

Command	Description
`meterInfo`	This command is used to send meter information to a host.

Table 4.8 `meters` Class Commands Originated By Host

Command	Description
`getMeterInfo`	This command is used to request meter information from an EGM.
`setMeterSub`	This command is used to set a meter subscription with an EGM.
`getMeterSub`	This command is used to request a host's current meter subscription from an EGM.
`clearMeterSub`	This command is used to remove a host's meter sub-scription from an EGM.

4.4.3 Events

Table 4.9 `meters` Class Events

Event
Meter Subscription Set
Meter Subscription Cleared

4.4.4 Meters

There are no meters associated with the activity in the `meters` class. The `meters` class is used to deliver meters associated with the activity of other classes.

4.5 auditMeters

The `auditMeters` class is used to identify whether an EGM supports an audit meter subscription and, if an audit meter subscription is supported, which host can set the audit meter subscription.

There are no meters, statuses, profiles, or logs associated with the `auditMeters` class.

4.5.1 Commands

There are no commands associated with the `auditMeters` class.

4.5.2 Events

There are no events associated with the `auditMeters` class.

4.5.3 Meters

There are no meters associated with the activity in the `auditMeters` class. The `meters` class is used to set meter subscriptions and deliver meters.

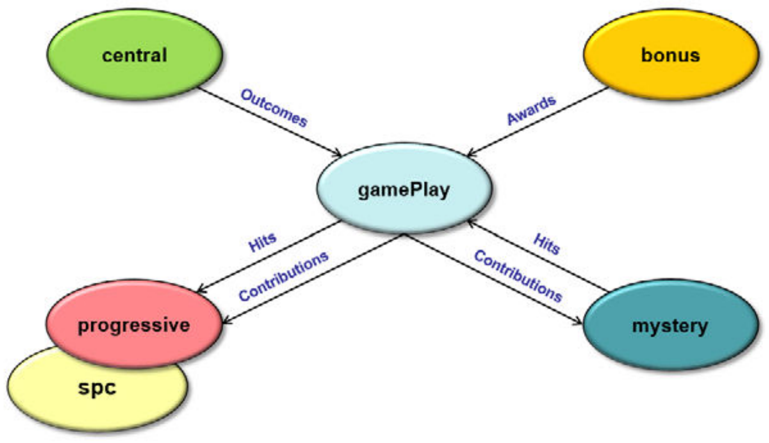
CHAPTER 5

Game Play Classes

5.1 Introduction

In addition to the `gamePlay` class itself, there are five additional classes associated with game play. These classes support advanced game play features such as progressive jackpots, promotional bonus awards, and central determination.

The illustration below demonstrates the relationship among the different game play classes.



The `central` class is used when `gamePlay` devices need central determination outcomes.

The `progressive` class is used to link games to progressive jackpot controllers. Contributions and jackpot hits are reported to the jackpot controllers.

The `spc` class can be layered on top of the `progressive` class when internal standalone progressive controllers are used to manage jackpots rather than external jackpot controllers. A host can configure the jackpots and monitor jackpot values through the `spc` class.

The `bonus` class is used to award promotional bonuses to players. Three types are available: standard, wager match, and jackpot multipliers.

Lastly, the `mystery` class is used to link games to mystery jackpot controllers. Contributions are reported to the jackpot controller; mystery jackpot awards are made by the controller.

5.2 gamePlay

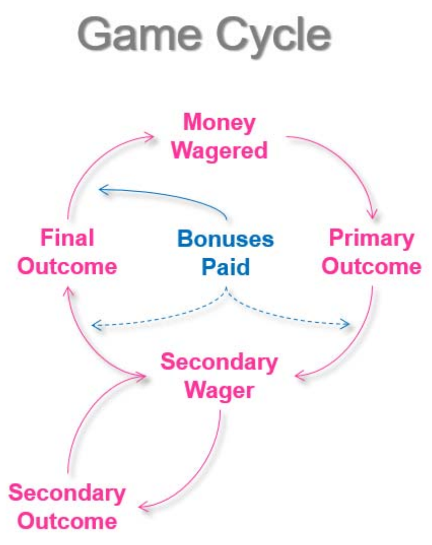
Benefit: The set of active games and denominations can be configured remotely by a host.

The primary purpose of the `gamePlay` class is to monitor game play activity on the EGM — specifically, meters and events associated with game play. With the `gamePlay` class you can also manage the set of active games and denominations on the EGM.

The `gamePlay` class uses all four G2S building blocks: status, profile, meters, and logs. There are two logs: the game recall log and the game outcome log. The game recall log is used with all types of games. It contains wager and win information for each game cycle. The game outcome log is only used with certain types of games. It contains internal game-specific information, such as the cards dealt in a poker game.

5.2.1 Game Cycles

G2S uses a standardized model for game cycles. The image below provides a visual representation of the game cycle.



5.2.2 Commands

Table 5.1 `gamePlay` Class Commands Originated by EGM.

Command	Description
	None.

Benefit: Game recall log includes detailed information about each game that was played and the win levels that were hit.

Table 5.2 `gamePlay` Commands Originated by Host

Command	Description
`setGamePlayState`	This command is used to enable or disable the device.
`getGamePlayStatus`	This command is used to request the current status information of the device.
`getGamePlayProfile`	This command is used to request the configuration information for the device.
`setActiveDenoms`	This command is used to set the list of active denominations for a game.
`getGameDenoms`	This command is used to request the list of denominations supported by a game.
`getRecallLogStatus`	This command is used to request the current status of the game recall log.
`getRecallLog`	This command is used to request the contents of the game recall log.
`getOutcomeLogStatus`	This command is used to request the current status of the game outcome log.
`getOutcomeLog`	This command is used to request the contents of the game outcome log.

5.2.3 Events

Table 5.3 `gamePlay` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host

Table 5.3 `gamePlay` Class Events

Event
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
All Device Faults Cleared
Primary Game Escrow
Primary Game Failed
Primary Game Started
Wager Changed
Primary Game Ended
Secondary Game Choice
Secondary Game Escrow
Secondary Game Failed
Secondary Game Started
Secondary Game Ended
Game Result
Game Ended
Game Idle
Active Denominations Changed
Device Fault
Initial Outcome Logged
Initial Outcome Updated
Final Outcome Logged
Extended Play Round
RTP Not Valid
RTP Valid
RTP Changed

5.2.4 Meters

Benefit: Primary game meters as well as secondary game (double-up) meters.

Table 5.4 `gamePlay` Class Performance Meters

Meters
Total amount wagered.
EGM-paid base paytable win.
Hand-paid base paytable win.
EGM-paid progressive win.
Hand-paid progressive win.
Number of primary games won by the player.
Number of primary games lost by the player.
Number of primary games where there were no credits won or lost.
Number of primary games where the game failed to complete.
Initial amounts wagered on secondary game cycles.
Final amounts won on secondary game cycles.
Count of secondary games won by the player.
Count of secondary games lost by the player.
Count of secondary games where there were no credits won or lost.
Count of secondary games where the game failed to complete.
Weighted theoretical payback amount.
Weighted theoretical payback percentage.
Count of extended play rounds.

Benefit: Theoretical win meters are calculated based on the actual wager categories played.

Table 5.5 `gamePlay` Class Wager Category Meters

Wager Category Meters
Amount wagered for the wager category.
Total count of games played for the wager category.
Theoretical payback percentage for the wager category.

Table 5.6 `gamePlay` Class Denomination Meters

Denomination Meters
Amount wagered for the denomination.
Total count of games played for the denomination.
Weighted theoretical payback amount for the denomination.
Weighted theoretical payback percentage for the denomination.

5.3 gameTheme

The `gameTheme` class is used to configure theme-related options that might be shared by multiple devices within the `gamePlay` class.The `gameTheme` class is a multi-device class.

The `gameTheme` class uses only one of the four G2S building blocks: profile. There are no meters, statuses, or logs associated with the `gameTheme` class.

5.3.1 Commands

Table 5.7 `gameTheme` Class Commands Originated by EGM

Command	Description
`gameThemeProfile`	This command is used to report the current profile information for a device.

Table 5.8 `gameTheme` Class Commands Originated by Host

Command	Description
`getGameThemeProfile`	This command is used to request the current profile information for a device.

5.3.2 Events

Table 5.9 gameTheme Class Events

Event
Device Configuration Changed by Host

5.3.3 Meters

There are no meters associated with the `gameTheme` class.

5.4 central

The `central` class is used to link a `gamePlay` device to a central determination host. The class includes commands for obtaining game outcomes from the central determination host.

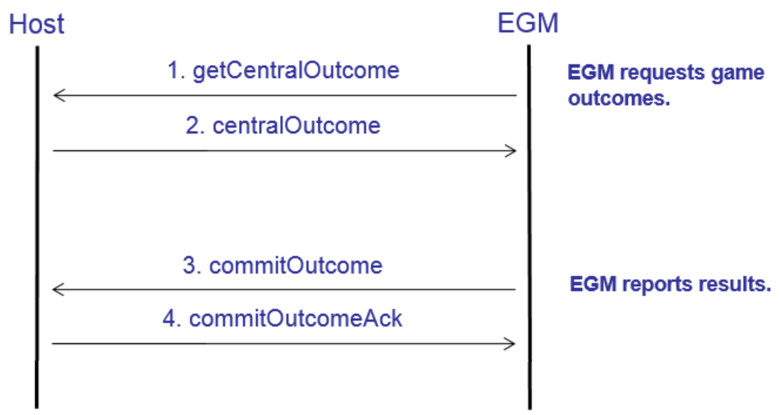
The `central` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters associated with the `central` class.

5.4.1 Command Sequence

The sequence diagram below illustrates how commands in the `central` class are intended to be used.

First, the EGM uses the `getCentralOutcome` command to request a set of game outcomes from the host. The host responds with the appropriate information via the `centralOutcome` command.

After the game outcomes have been used, the EGM reports the usage back to the host with the `commitOutcome` command and the host acknowledges this with the `commitOutcomeAck` command.



5.4.2 Commands

Table 5.10 `central` Class Commands Originated by EGM

Command	Description
`getCentralOutcome`	This command is used to request a set of outcomes from the host.
`commitOutcome`	This command is used to report the usage of a game outcome to the host.

Table 5.11 `central` Class Commands Originated by Host

Command	Description
`getCentralProfile`	This command is used to request the current configuration of a `central` device.
`setCentralState`	This command is used to enable or disable the use of a `central` device by the EGM.
`getCentralStatus`	This command is used to request the current status of a `central` device.
`getCentralLogStatus`	This command is used to request the current status of the class-level log.
`getCentralLog`	This command is used to request the contents of the class-level log.

5.4.3 Events

Table 5.12 `central` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Outcome Requested

Table 5.12 `central` Class Events

Event
Outcome Fault
Outcome Received
Outcome Committed

5.4.4 Meters

There are no meters associated with the `central` Class.

5.5 progressive

Benefit: Mappings between games and progressive jackpots can be configured remotely.

The `progressive` class is used to manage contributions and awards associated with progressive jackpots. The `progressive` class includes commands for broadcasting progressive jackpot values, reporting progressive wins, and awarding jackpots. Wins are determined by the EGM.

The `progressive` class supports multiple progressive controllers and multiple jackpot levels within each controller. Contribution meters are used to track qualifying wagers to each jackpot controller.

The `progressive` class uses all four G2S building blocks: status, profile, meters, and logs.

5.5.1 Command Sequence

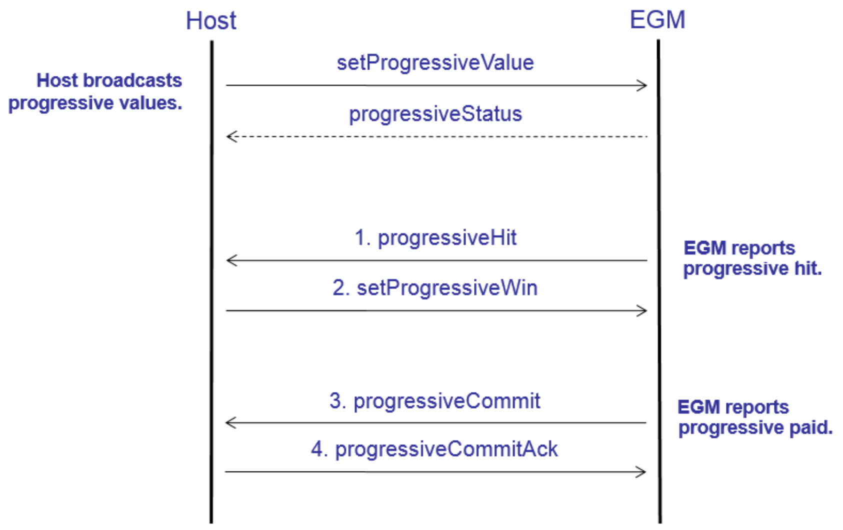
Benefit: Different win levels within a game can be mapped to different jackpot controllers – for example, a wide-area controller and a local-area controller.

The sequence diagram below illustrates how commands in the `progressive` class are intended to be used.

The host broadcasts current jackpot values to the EGM with the `setProgressiveValue` command. The EGM responds with the `progressiveStatus` command. The `setProgressiveValue` command can be sent via multicast.

When a progressive hit occurs, the EGM reports the hit via the `progressiveHit` command. The host acknowledges this with the `setProgressiveWin` command. The host can adjust the jackpot value as needed in the `setProgressiveWin` command.

After the jackpot has been paid, the EGM reports the payment with the `progressiveCommit` command. The host acknowledges the payment with the `progressiveCommitAck` command.



5.5.2 Commands

Table 5.13 `progressive` Class Commands Originated by EGM

Command	Description
`progressiveHit`	This command is used to notify the host that a progressive jackpot has been hit.
`progressiveCommit`	This command is used to report that a progressive jackpot has been paid.
`getProgressiveHostInfo`	This command can be used to query the host for progressive controller identifiers and jackpot levels.

Table 5.14 `progressive` Class Commands Originated by Host

Command	Description
`setProgressiveState`	This command is used to enable or disable a `progressive` device.
`setProgressiveValue`	This command is used to send the current value of one or more progressive levels to an EGM.
`setProgressiveLockout`	This command is used to request that the `progressive` device lock the EGM.

Table 5.14 `progressive` Class Commands Originated by Host

Command	Description
`getProgressiveProfile`	This command is used to request the current configuration of a `progressive` device.
`getProgressiveStatus`	This command is used to request the current status for a `progressive` device.
`getProgressiveLogStatus`	This command is used to request the current status of the class-level log.
`getProgressiveLog`	This command is used to request the contents of the class-level log.

5.5.3 Events

Table 5.15 `progressive` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Progressive Money Wagered
Progressive Hit
Progressive Hit Ack
Progressive Hit Committed
Progressive Hit Acknowledged
Progressive Failed

5.5.4 Meters

Table 5.16 `progressive` Class Transfer Meters

Meters
Value of cashable credits transferred in.
Value of promo credits transferred in.
Value of non-cashable credits transferred in.
Count of transfers in.

Benefit: Contribution meters report the qualifying wagers towards jackpots.

Table 5.17 `progressive` Class Contribution Meters

Meters
Money wagered against games that contribute to a `progressive` controller.
Count of the wagers against games that contribute to a `progressive` controller.

5.6 spc

Benefit: Standalone progressive jackpot controllers can be monitored and configured remotely.

The `spc` class is used to monitor and configure standalone progressive controllers. Each device in the `spc` class is linked to a device within the `progressive` class. Hits are reported through the `progressive` class; jackpot values and resets are reported through the `spc` class. Conventional progressive jackpots as well as mystery progressive jackpots are supported.

The `spc` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters associated with the `spc` class.

5.6.1 Commands

Table 5.18 `spc` Class Commands Originated by EGM

Command	Description
`spcLevelReset`	This command is used to report the reset of a standalone progressive level.

Table 5.19 `spc` Class Commands Originated by Host

Command	Description
`getSpcStatus`	This command is used to request the current status of the `spc` device.
`setSpcState`	This command is used to enable or disable the `spc` device.
`getSpcProfile`	This command is used to request the current configuration for the `spc` device.
`getSpcLogStatus`	This command is used to request the current status of the class-level log.
`getSpcLog`	This command is used to request the contents of the class-level log.

5.6.2 Events

Table 5.20 `spc` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Level Initialized
Level Adjusted
Level Update
Level Hit
Level Reset
Level Reset Acknowledged

5.6.3 Meters

There are no meters associated with the `spc` class.

5.7 bonus

Benefit: Three types of player bonuses are supported: standard, wager match, and jackpot multiplier.

The `bonus` class is used to manage the process of awarding external bonuses to an EGM. Wins and prize values are determined by the host. Examples include welcome-back bonuses, birthday bonuses, wager match, win multipliers, consolation prizes, etc. An EGM can be configured to communicate with multiple bonus controllers. Multiple bonus devices may be active in an EGM. Prizes can be awarded as cashable, promotional, or non-cashable credits.

The `bonus` class uses all four G2S building blocks: status, profile, meters, and transaction logs.

5.7.1 Command Sequence

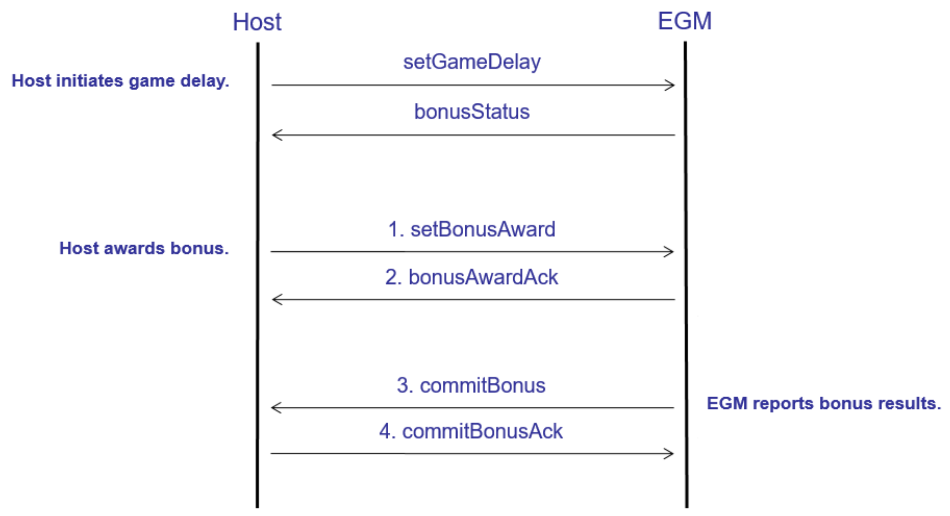
The sequence diagram below illustrates how commands in the `bonus` class are intended to be used.

First, the host initiates a game delay with the `setGameDelay` command (not always necessary). This creates a window of opportunity at the end of each game cycle for the host to award bonuses. The EGM responds with the `bonusStatus` command.

Benefit: The EGM manages the award of wager match and jackpot multiplier bonuses based on parameters set by the host.

When it's time to award a bonus, the host sends the `setBonusAward` command. The EGM acknowledges this with the `bonusAwardAck` command. The EGM will wait until the appropriate time to pay the bonus.

After the bonus has been paid, the EGM reports final results to the host with the `commitBonus` command. The EGM must retry the `commitBonus` command until the host acknowledges with the `commitBonusAck` command. When wager-match and win-multiplier sessions are enabled, the `commitBonus` command is not sent until the bonus sessions ends.



5.7.2 Commands

Table 5.21 `bonus` Class Commands Originated by EGM

Command	Description
`commitBonus`	The EGM uses this command to report that a bonus was paid.

Table 5.22 `bonus` Class Commands Originated by Host

Command	Description
`setBonusState`	This command is used to enable or disable a `bonus` device.
`setBonusLockOut`	This command is used to request that the `bonus` device lock the EGM.
`setGameDelay`	This command is used to set the current game delay parameters of the EGM.
`getBonusStatus`	This command is used to request the current status of a `bonus` device.
`getBonusProfile`	This command is used to request the current configuration of a `bonus` device.
`bonusActivity`	This command is used as a heartbeat message to verify that the host can still communicate with the `bonus` device.

Table 5.22 bonus Class Commands Originated by Host

Command	Description
`setBonusAward`	This command is used to authorize an EGM to award a bonus.
`cancelBonusAward`	This command is used to cancel a pending bonus award.
`setBonusMessage`	This command is used to send a message for the EGM to display to the player.
`getBonusLogStatus`	This command is used to request the current status of the class-level log.
`getBonusLog`	This command is used to request the contents of the class-level log.
`skipGameDelay`	This command is used to request that the EGM exit game delay, allowing the next game cycle to start.

5.7.3 Events

Table 5.23 bonus Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Game Delay Period Started
Game Delay Period Ended
Bonus Award Pending
Bonus Award Paid
Bonus Award Failed
Bonus Award Cancelled
Bonus Award Acknowledged

Table 5.23 `bonus` Class Events

Event
Bonus Host Communications Lost
Bonus Host Communications Restored
Wager Match Payment Method Not Available
Jackpot Multipliers Started
Display Limit Exceeded
Wager Match Limit Exceeded
Partial Bonus Award

5.7.4 Meters

Table 5.24 `bonus` Class Meters

Meters
Value of cashable credits transferred in.
Value of promo credits transferred in.
Value of non-cashable credits transferred in.
Count of transfers in.

5.8 mystery

Benefit: Mappings between games and mystery jackpots can be configured remotely.

The mystery class is used to manage the configuration and operation of mystery jackpots. Commands for broadcasting mystery jackpot values and for awarding mystery jackpots are included. Like the bonus class, wins are determined by the host and game-delay can be enabled. Like the progressive class, the mystery class supports contributions, multiple controllers, and multiple jackpot levels.

The mystery class uses all four G2S building blocks: status, profile, meters, and logs.

5.8.1 Commands

Table 5.25 mystery Class Commands Originated by EGM

Command	Description
gameResult	This command is used to send game results to the host during game-delay periods.
mysteryCommit	This command is used to notify the host that a mystery jackpot has been paid.

Table 5.26 mystery Class Commands Originated by Host

Command	Description
getMysteryProfile	This command is used to request the current configuration of the mystery device.
getMysteryStatus	This command is used to request the current status of the mystery device.
setMysteryState	This command is used to enable or disable the functionality associated with the mystery device.
setMysteryLockOut	This command is used to request that the device lock the EGM.
setMysteryGameDelay	This command is used to set the current game-delay parameters for the device.
skipMysteryGameDelay	This command is used to instruct the EGM to exit game-delay for the current game cycle.
setMysteryValue	This command is used to send the current value of one or more mystery jackpot levels to the EGM.

Table 5.26 `mystery` Class Commands Originated by Host

Command	Description
`setMysteryMessage`	This command is used to send a message for the EGM to display to the player.
`mysteryAward`	This command is used to authorize the EGM to award a mystery jackpot to the player.
`getMysteryLogStatus`	This command is used to request the current status of the class-level log.
`getMysteryLog`	This command is used to request the contents of the class-level log.

5.8.2 Events

Table 5.27 `mystery` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Mystery Jackpot Money Wagered
Game-Delay Period Started
Game-Delay Period Ended
Mystery Jackpot Award Pending
Mystery Jackpot Awarded
Mystery Jackpot Award Failed
Mystery Jackpot Award Acknowledged
Host Communications Lost
Host Communications Restored

5.8.3 Meters

Table 5.28 `mystery` Class Transfer Meters

Transfer Meters
Value of cashable credits transferred in.
Value of promo credits transferred in.
Value of non-cashable credits transferred in.
Count of transfers in.

Benefit: Contribution meters report the qualifying wagers towards jackpots.

Table 5.29 `mystery` Class Contribution Meters

Contribution Meters
Money wagered against games that contribute to a `mystery` controller.
Count of the wagers against games that contribute to a `mystery` controller.

5.9 sign

The `sign` class is used to manage signs that display progressive and mystery jackpot values. It can also be used to manage the display of recent jackpot wins.

Jackpot values are broadcast by the `progressive` and `mystery` classes. As the player changes games, different jackpot levels linked to different games become eligible for display. The `sign` class assures that the jackpot levels linked to the currently selected game are displayed to the player in the proper positions on the sign.

The `sign` class uses two of the four G2S building blocks: status and profile. There are no meters or logs with the `sign` class.

5.9.1 Commands

Benefit: Contribution meters report the qualifying wagers towards jackpots.

Table 5.30 `sign` Class Commands Originated by EGM.

Command	Description
	None.

Table 5.31 `sign` Class Commands Originated by Host

Command	Description
`getSignProfile`	This command is used to request the current configuration of the `sign` device.
`getSignStatus`	This command is used to request the current status of the `sign` device.
`setSignState`	This command is used to enable or disable the functionality associated with the `sign` device.
`setDisplayText`	This command is used to override the information being displayed at a specific display position with specific text.
`setJackpotHits`	This command is used to send lists of jackpot hits to the EGM.

5.9.2 Events

Table 5.32 `sign` Class Events

Events
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Information Displayed to Player Has Changed

5.9.3 Meters

There are no meters associated with the `sign` Class.

5.10 cabinet

Benefit: Processor resets and master resets can be triggered remotely.

The `cabinet` class is responsible for the physical housing and security of the EGM. Additionally, the `cabinet` class includes commands for enabling, disabling, and locking the EGM, disabling money in, disabling game play, setting local time zone changes, and setting operating hours.

The `cabinet` class uses three of the four G2S building blocks: status, profile, and meters. There are no logs in the `cabinet` class.

5.10.1 Commands

Table 5.33 `cabinet` Class Commands Originated by EGM

Command	Description
`masterResetStatus`	This command is used to report the status of a master reset.

Table 5.34 `cabinet` Class Commands Originated by Host

Benefit: Time zone offsets can be configured into the EGM to manage the transition to/from daylight savings time.

Benefit: Operating hours can be configured into the EGM to automatically enable/disable the EGM at prescribed times of day.

Command	Description
`getCabinetStatus`	This command is used to request the current status of the `cabinet` device.
`getCabinetProfile`	This command is used to request the current configuration of the `cabinet` device.
`setCabinetState`	This command is used to enable or disable the EGM.
`setCabinetLockOut`	This command is used to lock the EGM.
`setDateTime`	This command is used to set the current date and time of the EGM.
`getDateTime`	This command is used to request the current date and time of the EGM.
`resetProcessor`	This command is used to remotely restart the EGM.
`setTimeZoneOffsets`	This command is used to control when an EGM must change time zone offsets.
`getTimeZoneOffsets`	This command is used to request the list of current time zone offset changes for the EGM.
`setOperatingHours`	This command is used to set the operating hours for the EGM.

Table 5.34 `cabinet` Class Commands Originated by Host

Command	Description
`getOperatingHours`	This command is used to request the operating hours of the EGM.
`masterReset`	This command is used to initiate a master reset, returning the EGM to 'Factory' conditions.
`authorizeMasterReset`	This command is used to inform the EGM that a host authorizes or denies the initiation of a master reset.
`cancelMasterReset`	This command is used to cancel a master reset request that was previously sent to the EGM.
`getMasterResetStatus`	This command is used to request the current status of a master reset.
`resetAuditPending`	This command is used to re-enable the EGM after a win that exceeded the audit limit.
`sealLogicDoor`	This command is used to re-enable the EGM after the logic area was accessed.

5.10.2 Events

Table 5.35 `cabinet` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested By Host
EGM Lock Not Requested by Host
Host Disabled Game Play
Host Enabled Game Play
Host Disabled Money In

Table 5.35 `cabinet` Class Events

Event
Host Enabled Money In
Communications Issue Disabled EGM
EGM Disabled Via Operator Menu
Device Action Disabled EGM
Host Command Disabled EGM
EGM Enabled
Operator Menu Activated
Demo Mode Activated
Meters/Audit Mode Activated
EGM Locked Via Operator Menu
Device Action Locked EGM
Host Action Locked EGM
Service Lamp On
Service Lamp Off
Logic Door Open
Logic Door Closed
Auxiliary Door Open
Auxiliary Door Closed
Cabinet Door Open
Cabinet Door Closed
General Cabinet Fault
Video Display Error
Non-Volatile Storage Fault
General Memory Fault
All Cabinet Faults Cleared
Game Change Committed
Date/Time Changed
Cash-Out Button Pressed
Power-Off Logic Door Open
Power-Off Auxiliary Door Open
Power-Off Cabinet Door Open

Table 5.35 `cabinet` Class Events

Event
Operator Reset Cabinet
Life-To-Date Meters Reset
Non-Volatile Storage Cleared
Backup Battery Low
EGM Power Up/Restart
Hard Meters Disconnected
Hard Meters Reconnected
Cabinet Reset Started
Remote Cabinet Reset
EGM Idle
EGM Not Idle
Current Locale Changed
Time Zone Offset Changed
EGM Occupied
Occupancy Suspended
EGM Not Occupied
Host Sets Operating Hours
EGM Sets Operating Hours
Operating Hours Expired
Operating Hours Enabled
Master Reset Pending
Master Reset Authorized
Master Reset Started
Master Reset Aborted
Master Reset Cancelled
Master Reset Waiting for Authorization
Host Authorization Received
Master Reset Time-Out
Audit Win Pending
Audit Win Not Pending
Significant Win

Table 5.35 `cabinet` Class Events

Event
Logic Seal Broken
Logic Door Sealed
Illegal Logic Door Open
Illegal Auxiliary Door Open
Illegal Cabinet Door Open
Game Combo Selected

5.10.3 Meters

Benefit: Occupancy meters track the amount of time that an EGM is actually in use.

Table 5.36 `cabinet` Class Meters

Meters
Value of cashable credit meter.
Value of promotional credit meter.
Value of non-cashable credit meter.
Total cashable credits wagered on the EGM.
Total promotional credits wagered on the EGM.
Total non-cashable credits wagered on the EGM.
Total EGM-paid base paytable win.
Total hand-paid base paytable win.
Total EGM-paid progressive win.
Total hand-paid progressive win.
Total EGM-paid external bonus awards transferred as win.
Total hand-paid external bonus awards transferred as win.
Total EGM-paid external bonus awards transferred as non-win.
Total hand paid external bonus awards transferred as non-win.
Total cashable credits paid out via currency.
Total promotional credits paid out via currency.
Total non-cashable credits paid out via currency.
Total hand-paid cancel credits.

Table 5.36 `cabinet` Class Meters

Meters
Number of games since initialization.
Number of games since the last power up.
Number of games since the last cabinet door closed.
Number of legal and illegal logic door opens.
Number of legal and illegal auxiliary door opens.
Number of legal and illegal cabinet door opens.
Number of power-off logic door open detections.
Number of power-off auxiliary door open detections.
Number of power-off cabinet door open detections.
Amount of time (in milliseconds) that the EGM was occupied.
Number of errors detected while reading IDs.
Total paytable win amounts that resulted in the generation of the Audit Win Pending event.
Number of Audit Win Pending events.
Total paytable win amounts that resulted in the generation of the Significant Win event.
Number of Significant Win events.
Number of times the logic seal was broken.
Number of illegal logic door opens.
Number of illegal auxiliary door opens.
Number of illegal cabinet door opens.
Number of games played while the MJT bonus mode was active.
Number of games awarded an MJT bonus while the MJT bonus mode was active.
Total amount of MJT bonuses paid.
Number of games played while the Wager Match bonus mode was active.
Total amount of Wager Match bonuses paid.

5.11 cashout

The `cashout` class is used by the host to initiate a cash-out on an EGM. The net affect is similar to the player pressing the cash-out button the EGM. However, the commands also allow the host to initiate partial cash-outs. The `cashout` class is a multi-device class.

The `cashout` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the `cashout` class.

5.11.1 Commands

Table 5.37 `cashout` Class Commands Originated by EGM

Command	Description
`commitCashoutRequest`	This command is used to report the results of a cash-out request.

Table 5.38 `cashout` Class Commands Originated by Host

Command	Description
`setCashoutState`	This command is used to enable or disable host-initiated cash-out activity for an EGM.
`getCashoutStatus`	This command is used by the host to request the current status of a `cashout` device from the EGM.
`getCashoutProfile`	This command is used by the host to request the current profile for a `cashout` device from the EGM.
`initiateCashoutRequest`	This command is used to request the EGM to initiate a cash-out sequence.
`cancelCashoutRequest`	This command is used to cancel a pending cash-out that was requested by the `initiateCashoutRequest` command.
`getHostCashoutLogStatus`	This command is used by the host to request the current status of the class-level host-initiated `cashout` transaction log from an EGM.
`getHostCashoutLog`	This command is used by the host to request the contents of the class-level host-initiated `cashout` transaction log from an EGM.

5.11.2 Events

Table 5.39 `cashout` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Cash-Out Requested by Host
Cash-Out Completed
Cash-Out Request Failed
Cash-Out Commit Command Acknowledged
Cash-Out Cancelled

5.11.3 Meters

There are no meters associated with the `cashout` class.

CHAPTER 6

Money Movement Classes

6.1 Introduction

There are many ways to move money on and off an EGM including currency, handpays, vouchers (tickets), and electronic transfers. In total, there are nine money movement classes in G2S. Of the nine classes, four classes deal with physical currency: coin acceptor, coin hopper, note acceptor, and note dispenser. Three classes deal with electronic transfers: smart cards, wagering account transfers (WAT), and direct funds transfers (DFT). The remaining two classes deal with handpays and vouchers.

All classes support cashable, promotional, and non-cashable credits. All classes that deal with physical currency support foreign currency exchange. Foreign currencies are converted to the EGM's base currency. Except for handpays and vouchers, multiple instances of each type of device can be supported by an EGM.

6.2 coinAcceptor

Benefit: Supports acceptance of foreign currencies; currency exchange rates can be configured remotely.

The `coinAcceptor` class includes commands and events related to coin acceptors, diverters, and drop boxes.

The `coinAcceptor` class uses three G2S building blocks: status, profile, and meters. Logs are not used with the `coinAcceptor` class. Meters are available in total or broken down by currency type (coin or token), currency code, and denomination.

6.2.1 Commands

Table 6.1 `coinAcceptor` Class Commands Originated by EGM

Command	Description
	None.

Table 6.2 `coinAcceptor` Class Commands Originated by Host

Command	Description
`setCoinAcceptorState`	This command is used to enable or disable a `coinAcceptor` device.
`getCoinAcceptorStatus`	This command is used to request the current status of a `coinAcceptor` device.
`getCoinAcceptorProfile`	This command is used to request the current configuration of a `coinAcceptor` device.

6.2.2 Events

Table 6.3 `coinAcceptor` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host

Table 6.3 `coinAcceptor` Class Events

Event
Device Configuration Changed by Operator
All Device Faults Cleared
Acceptor Jammed
Acceptor Fault
Diverter Fault
Coin Acceptor Lockout Malfunction
Inappropriate Coin Not Returned
Coin In Limit Reached
Drop Door Opened
Drop Door Closed
Coins Accepted
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Illegal Activity Detected
Power-Off Drop Door Opened
Illegal Drop Door Opened

6.2.3 Meters

Table 6.4 `coinAcceptor` Class Meters

Meters
Cashable value of currency accepted by the EGM.
Promotional value of currency accepted by the EGM.
Non-cashable value of currency accepted by the EGM.

Table 6.4 `coinAcceptor` Class Meters

Meters
Count of currency accepted by the EGM.
Cashable value of currency diverted to the drop box.
Promotional value of currency diverted to the drop box.
Non-cashable value of currency diverted to the drop box.
Count of currency diverted to the drop box.
Cashable value of currency diverted to the hopper.
Promotional value of currency diverted to the hopper.
Non-cashable value of currency diverted to the hopper.
Count of currency diverted to the hopper.
Count of drop door opens.
Count of power-off drop door opens.
Count of illegal drop door opens.

6.3 noteAcceptor

Benefit: Supports acceptance of foreign currencies; currency exchange rates can be configured remotely.

The `noteAcceptor` class includes commands and events related to note acceptors and stackers, including the acceptance of vouchers (tickets).

The `noteAcceptor` class uses all four G2S building blocks: status, profile, meters, and logs. Meters are available in total or broken down by currency type (note or scrip), currency code, and denomination.

6.3.1 Commands

Table 6.5 `noteAcceptor` Commands Originated by EGM

Command	Description
	None.

Table 6.6 `noteAcceptor` Commands Originated by Host

Command	Description
`setNoteAcceptorState`	This command is used to enable or disable a `noteAcceptor` device.
`getNoteAcceptorStatus`	This command is used to request the current status of a `noteAcceptor` device.
`getNotesAcceptedStatus`	This command is used to request the current status of the class-level log.
`getNotesAccepted`	This command is used to request the contents of the class-level log.
`getNoteAcceptorProfile`	This command is used to request the current configuration of a `noteAcceptor` device.

6.3.2 Events

Table 6.7 `noteAcceptor` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM

Table 6.7 `noteAcceptor` Class Events

Event
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
All Device Faults Cleared
Acceptor Jammed
Acceptor Fault
Stacker Removed
Stacker Inserted
Stacker Full
Stacker Jammed
Stacker Fault
Note Returned
Voucher Returned
Note or Voucher Rejected
Validator Totals Reset
Stacker Door Opened
Stacker Door Closed
Note Stacked
Voucher Stacked
Note In Escrow
Stacker Nearly Full
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Illegal Activity Detected
Power-Off Stacker Door Open

Table 6.7 `noteAcceptor` Class Events

Event
Power-Off Stacker Removed
Excessive Note or Voucher Rejected
Illegal Stacker Door Opened

6.3.3 Meters

Table 6.8 `noteAcceptor` Class Meters

Meters
Cashable value of currency accepted by the EGM.
Promotional value of currency accepted by the EGM.
Non-cashable value of currency accepted by the EGM.
Count of currency accepted by the EGM.
Cashable value of currency diverted to the stacker.
Promotional value of currency diverted to the stacker.
Non-cashable value of currency diverted to the stacker.
Count of currency diverted to the stacker.
Cashable value of currency diverted to the note dispenser.
Promotional value of currency diverted to the note dispenser.
Non-cashable value of currency diverted to the note dispenser.
Count of currency diverted to the note dispenser.
Count of stacker door opens.
Count of power-off stacker door opens.
Count of illegal stacker door opens.
Count of stacker removals.
Count of power-off stacker removals.

6.4 hopper

Benefit: Supports acceptance of foreign currencies; currency exchange rates can be configured remotely.

The `hopper` class includes commands and events related to coin hoppers.

The `hopper` class uses three G2S building blocks: status, profile, and meters. Logs are not used with the `hopper` class. Meters are available in total or broken down by currency type (coin or token), currency code, and denomination.

6.4.1 Commands

Table 6.9 `hopper` Commands Originated by EGM

Command	Description
	None.

Table 6.10 `hopper` Commands Originated by Host

Command	Description
`setHopperState`	This command is used to enable or disable a `hopper` device.
`getHopperStatus`	This command is used to request the current status of a `hopper` device.
`getHopperProfile`	This command is used to request the current configuration of a `hopper` device.

6.4.2 Events

Table 6.11 `hopper` Class Events

Events
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host

Table 6.11 `hopper` Class Events

Events
Device Configuration Changed by Operator
All Device Faults Cleared
Hopper Empty
Hopper Full
Hopper Jammed
Hopper Below High Water Mark
Hopper Above High Water Mark
Hopper Fault
Extra Coins Paid
Runaway Hopper
Dispenser Door Opened
Dispenser Door Closed
Coins Dispensed
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Illegal Activity Detected
Power-Off Dispenser Door Opened
Hopper Not Full
Illegal Dispenser Door Open

6.4.3 Meters

Table 6.12 `hopper` Class Meters

Meters
Cashable value of currency dispensed by the EGM.
Non-cashable value of currency dispensed by the EGM.
Non-cashable value of currency dispensed by the EGM.
Count of currency dispensed by the EGM.
Count of dispenser door opens.
Count of power-off dispenser door opens.
Count of illegal dispenser door opens.

6.5 noteDispenser

Benefit: Supports acceptance of foreign currencies; currency exchange rates can be configured remotely.

The `noteDispenser` class includes commands and events related to note dispensers.

The `noteDispenser` class uses all four G2S building blocks: status, profile, meters, and logs. Meters are available in total or broken down by currency type (note or scrip), currency code, and denomination.

6.5.1 Commands

Table 6.13 `noteDispenser` Commands Originated by EGM

Command	Description
	None.

Table 6.14 `noteDispenser` Commands Originated by Host

Command	Description
`setNoteDispenserState`	This command is used to enable or disable a `noteDispenser` device.
`getNoteDispenserStatus`	This command is used to request the current status of a `noteDispenser` device.
`getNotesDispensedStatus`	This command is used to request the current status of the class-level log.
`getNotesDispensedLog`	This command is used to request the contents of the class-level log.
`getNoteDispenserProfile`	This command is used to request the current configuration of a `noteDispenser` device.

6.5.2 Events

Table 6.15 `noteDispenser` Class Events

Events
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
All Device Faults Cleared
Dispenser Empty
Dispenser Full
Dispenser Jammed
Dispenser Below High Water Mark
Dispenser Above High Water Mark
Dispenser Below Low Water Mark
Dispenser Above Low Water Mark
Dispenser Fault
Dispenser Door Opened
Dispenser Door Closed
Note Dispensed
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Illegal Activity Detected
Power-Off Dispenser Door Opened

Table 6.15 `noteDispenser` Class Events

Events
Dispenser Not Full
Illegal Dispenser Door Open

6.5.3 Meters

Table 6.16 `noteDispenser` Class Meters

Meters
Cashable value of currency dispensed by the EGM.
Promotional value of currency dispensed by the EGM.
Non-cashable value of currency dispensed by the EGM.
Count of currency dispensed by the EGM.
Count of dispenser door opens.
Count of power-off dispenser door opens.
Count of illegal dispenser door opens.

6.6 handpay

Benefit: Handpays can be keyed off locally by an attendant or remotely by the host.

The `handpay` class is used to manage large wins and other payouts that cannot be made by the EGM, including cancelled-credits, external bonuses, and direct funds transfers.

The `handpay` class includes commands for reporting handpay requests and keying off those requests. Payments can made by an attendant or they can be made to vouchers, wagering accounts, smart cards, etc. Key-offs can be initiated locally by an attendant or remotely by the host.

The `handpay` class uses all four G2S building blocks: status, profile, meters, and logs.

6.6.1 Command Sequence

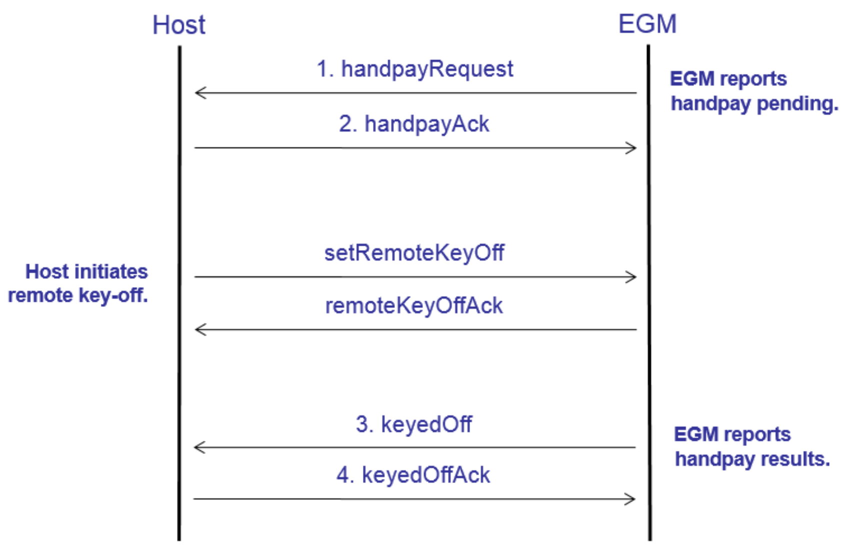
Benefit: Host can select the payment method for a handpay — attendant, ticket, wagering account, smart card, or credit meter.

The sequence diagram below illustrates how commands in the `handpay` class are intended to be used.

First, the EGM sends a `handpayRequest` command to the host. The EGM is reporting that a handpay is pending. The host responds with the `handpayAck` command.

Next, the host uses the `setRemoteKeyOff` command to tell the EGM how the handpay request should be paid — by an attendant, voucher, wagering account, smart card, or to the credit meter; locally or remotely. This step is optional. If the host says nothing, only local key-off options will be available. The EGM responds with the `remoteKeyOffAck` command.

Finally, the EGM sends the `keyedOff` command to the host, reporting the handpay results. The `keyedOff` command must be retried until acknowledged by the host with the `keyedOffAck` command.



6.6.2 Commands

Table 6.17 `handpay` Class Commands Originated by EGM

Command	Description
`handpayRequest`	This command is used to send a handpay request to a host.
`keyedOff`	This command is used to notify a host that a key-off has taken place.

Table 6.18 `handpay` Class Commands Originated by Host

Command	Description
`getHandpayProfile`	This command is used to request the current handpay configuration from an EGM.
`setHandpayState`	This command is used to enable or disable handpays at an EGM.
`getHandpayStatus`	This command is used to request the current status of handpays from an EGM.
`setRemoteKeyOff`	This command is used to initiate a remote key-off of a handpay or authorize local key-off types at an EGM.

Table 6.18 `handpay` Class Commands Originated by Host

Command	Description
`getHandpayLogStatus`	This command is used to request the current status of the handpay transaction log from an EGM.
`getHandpayLog`	This command is used to request the contents of the handpay transaction log from an EGM.

6.6.3 Events

Table 6.19 `handpay` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Handpay Pending
Handpay Request Acknowledged
Remote Key-Off Initiated
Handpay Keyed Off
Key-Off Acknowledged
Cancelled-Credit Request Cancelled

6.6.4 Meters

Table 6.20 `handpay` Class Meters

Meters
Value of cashable credits transferred out.
Value of promotional credits transferred out.

Table 6.20 `handpay` Class Meters

Meters
Value of non-cashable credits transferred out.
Count of transfers out.

6.7 voucher

The `voucher` class is used to manage the process of issuing and redeeming payment vouchers (tickets) at an EGM. The `voucher` class supports cashable, promotional, or non-cashable vouchers. The `voucher` class includes commands for requesting validation IDs, reporting issued vouchers and authorizing voucher redemptions. Validation IDs are supplied by the host. A seed value, which is included with each validation ID, is used to produce a manual authentication code.

The `voucher` class uses all four G2S building blocks: status, profile, meters, and logs.

6.7.1 Manual Authentication Codes

Benefit: Manual authentication codes are printed on tickets to support offline ticket validation.

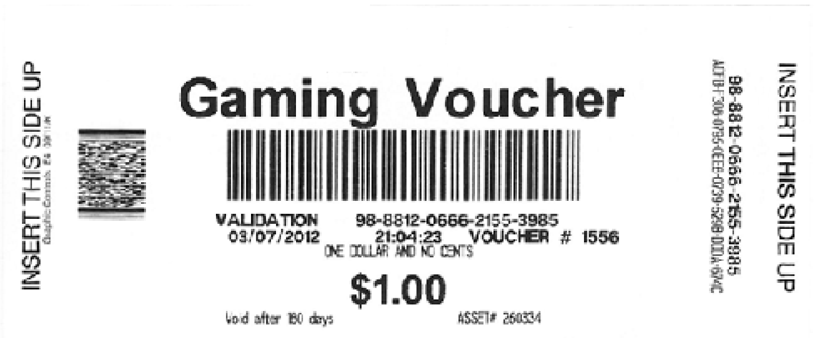
Manual authentication codes are designed so that voucher validation can occur even when the EGM is offline. With other protocols, the only way to find out whether the EGM issued the voucher is to go to the EGM and view the transaction logs. This is a very time-consuming process.

The G2S manual authentication code is a SHA1 hash of the EGM ID, validation ID, seed value, and voucher amount. The SHA1 hash is calculated across all four parameters and then converted into a 32-character string. The string is printed on the voucher.

The host system knows the EGM ID, validation ID, and seed value. The only thing it doesn't know is the amount. Thus, if a cashier enters the voucher amount, the host can recalculate the manual authentication code. This value can be compared to the value printed on the ticket.

6.7.2 voucher example

The illustration below is an example of a voucher with a manual authentication code. The code is printed on the right side of the voucher below the validation ID.



INSERT THIS SIDE UP

Gaming Voucher

VALIDATION 98-8812-0666-2155-3985
03/07/2012 21:04:23 VOUCHER # 1556
ONE DOLLAR AND NO CENTS

$1.00

Void after 180 days ASSET# 260334

98-8812-0666-2155-3985

INSERT THIS SIDE UP

The manual authentication code is also included in the PDF417 bar code printed on the left side of the voucher. The PDF417 bar code includes other useful information as well, such as the EGM ID, validation ID, and voucher amount. Instead of the cashier entering this information to perform manual authentication, the data can be read from the PDF417 bar code, making the process more efficient.

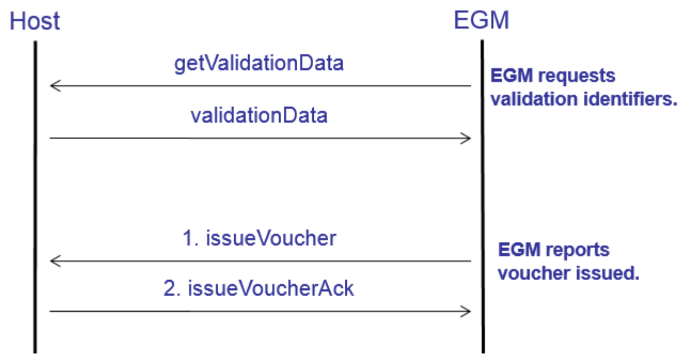
The validation ID is included in the interleaved 2-of-5 bar code in the center of the voucher.

6.7.3 Command Sequence

The sequence diagrams below illustrate how commands in the `voucher` class are intended to be used.

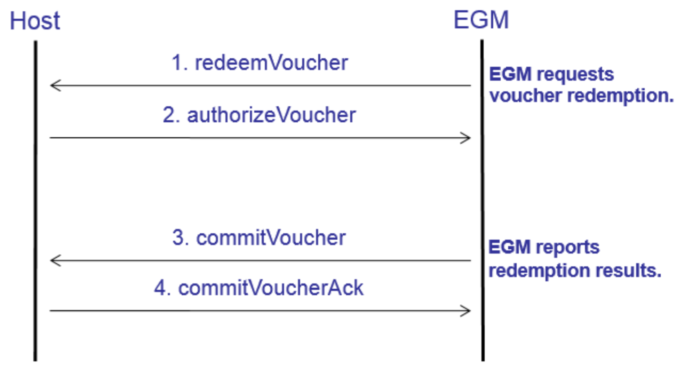
When issuing a voucher, first, the EGM sends a `getValidationData` command to the host to request the validation identifiers. The host responds with the `validationData` command. The EGM will stop issuing vouchers if validation identifiers are exhausted or expire.

Next, to report the issuance of a voucher, the EGM sends an `issueVoucher` command. The host acknowledges this with the `issueVoucherAck` command. The EGM must retry the `issueVoucher` command until the host acknowledges.



When redeeming a voucher, first, the EGM sends a `redeemVoucher` command to the host to request permission to redeem the voucher. The host responds with the `authorizeVoucher` command which contains the voucher amount and credit type.

Next, the EGM reports the redemption results with a `commitVoucher` command. The host acknowledges with a `commitVoucherAck` command. The EGM must retry the `commitVoucher` command until the host acknowledges.



6.7.4 Commands

Table 6.21 `voucher` Class Commands Originated by EGM

Command	Description
`getValidationData`	This command is used to request new validation IDs.
`issueVoucher`	This command is used to notify a host that a voucher has been issued.

Table 6.21 `voucher` Class Commands Originated by EGM

Command	Description
`redeemVoucher`	This command is used to send a voucher redemption request to a host.
`commitVoucher`	This command is used to report the results of a voucher redemption request.

Table 6.22 `voucher` Class Commands Originated by Host

Command	Description
`setVoucherState`	This command is used to enable or disable the voucher functionality.
`setVoucherLockout`	This command is used to request that the device lock the EGM.
`getVoucherStatus`	This command is used to request the current status of the voucher functionality.
`getVoucherProfile`	This command is used to request the current voucher configuration.
`getVoucherLogStatus`	This command is used by the host to request the current status of the voucher transaction log.
`getVoucherLog`	This command is used by the host to request the contents of the voucher transaction log.

6.7.5 Event Codes

Table 6.23 `voucher` Class Event Codes

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host

Table 6.23 `voucher` Class Event Codes

Event
Validation Identifier Data Expired
Validation Identifier Data Updated
Voucher Issued
Voucher Issue Command Acknowledged
Voucher Redemption Requested
Voucher Authorized
Voucher Redeemed
Voucher Rejected
Voucher Commit Command Acknowledged
Validation System Offline
Validation System Not Offline

6.7.6 Meters

Table 6.24 `voucher` Class Voucher Issued Meters

Meters - Voucher Issued
Value of issued cashable voucher.
Count of issued cashable vouchers.
Value of issued promotional voucher.
Count of issued promotional vouchers.
Value of issued non-cashable voucher.
Count of issued non-cashable vouchers.

Table 6.25 `voucher` Class EGM Issued Voucher Redeemed Meters

Meters - EGM Issued Voucher Redeemed
Value of redeemed cashable voucher.
Count of redeemed cashable vouchers.
Value of redeemed promotional voucher.
Count of redeemed promotional vouchers.

Table 6.25 `voucher` Class EGM Issued Voucher Redeemed Meters

Meters - EGM Issued Voucher Redeemed
Value of redeemed non-cashable voucher.
Count of redeemed non-cashable voucher.

Table 6.26 `voucher` Class System Issued Voucher Redeemed Meters

Meters - System Issued Voucher Redeemed
Value of redeemed cashable voucher.
Count of redeemed cashable vouchers.
Value of redeemed promotional voucher.
Count of redeemed promotional vouchers.
Value of redeemed non-cashable voucher.
Count of redeemed non-cashable voucher.

6.8 wat (wagering account transfer)

Benefit: Transfers can be initiated by the host or the EGM.

The `wat` class is used to manage the process of transferring funds between an EGM and a player account on a host system. Funds can be cashable, promotional, or non-cashable.

The user interface for wagering account transfers can be implemented in the EGM or in an external device managed by the host system.

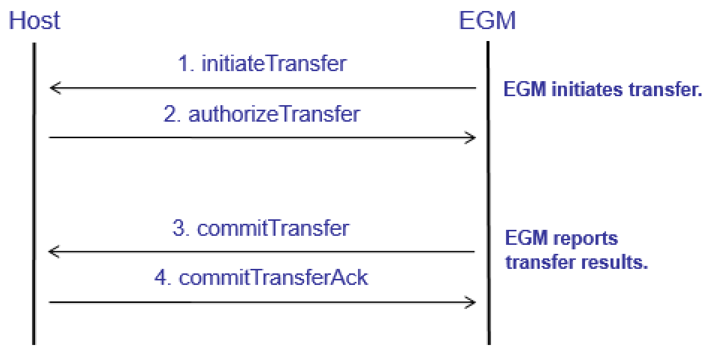
The `wat` class uses all four G2S building blocks: status, profile, meters, and logs.

6.8.1 Command Sequence

The sequence diagram below illustrates how commands in the `wat` class are intended to be used to make a wagering account transfer. The following sections discuss the two modes of operation that are available for triggering the transfer.

First, the EGM sends an `initiateTransfer` command to the host. The host responds with the `authorizeTransfer` command. The host specifies the transfer amount. The EGM may reduce this amount.

Second, the EGM then sends a `commitTransfer` command to the host reporting the results of the transfer. To conclude, the host responds with a `commitTransferAck` command. The EGM must retry `commitTransfer` until acknowledged via `commitTransferAck`.



6.8.2 Interface Mode

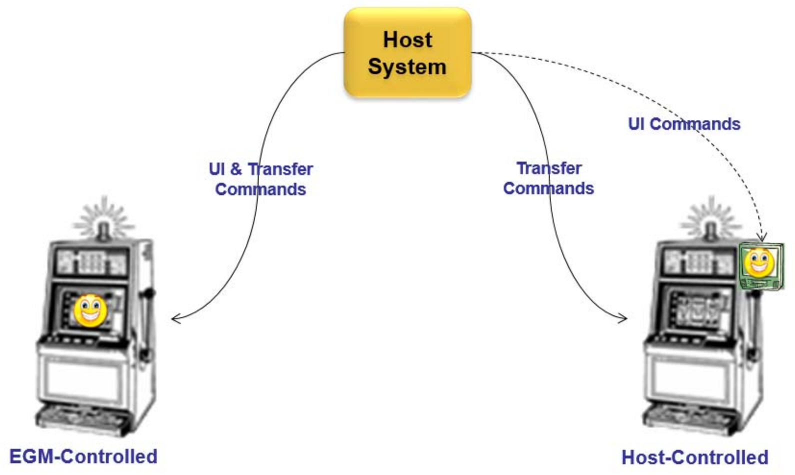
The `wat` class supports two modes of operation for triggering wagering account transfers: EGM-controlled and host-controlled.

The illustration below displays the relationship between host and EGM with each mode of operation. When EGM-controlled, the commands used to manage the user-interface, as well as the transfers, are sent through G2S. When host-controlled, only the commands used to manage the transfers are sent through G2S. The commands used to manage the user-interface are proprietary; typically, the user-interface is implemented on a separate external device.

When EGM-controlled, the user interface can be built, tested, and approved on the EGM and no external device is required.

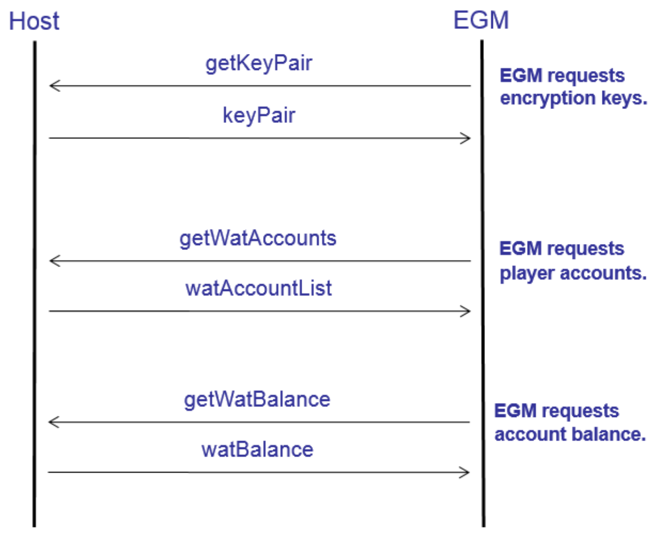
When host-controlled, the user interface is implemented in an external device managed by the host system. The external device must be built, tested, and approved separately from the EGM.

The protocol doesn't dictate which method is used. That is decided by implementers of the protocol.



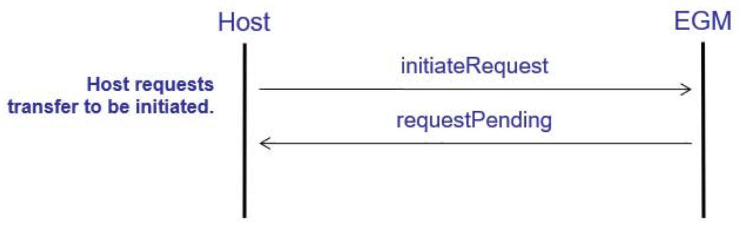
6.8.3 EGM-Controlled Command Sequence

The sequence diagram below illustrates how commands in an EGM-Controlled environment work.


Host
EGM
getKeyPair
keyPair
EGM requests encryption keys.
getWatAccounts
watAccountList
EGM requests player accounts.
getWatBalance
watBalance
EGM requests account balance.

6.8.4 Host-Controlled Command Sequence

The sequence diagram below illustrates how commands in a Host-Controlled environment work. The host simply sends a command to trigger the wagering account transfer.



6.8.5 Commands

Table 6.27 `wat` Class Commands Originated by EGM

Command	Description
`getWatAccounts`	This command is used in EGM-controlled mode to retrieve all wagering accounts for a player.
`getWatBalance`	This command is used in EGM-controlled mode to request the current balance in a wagering account.

Table 6.27 `wat` Class Commands Originated by EGM

Command	Description
`initiateTransfer`	This command is used to initiate a transfer.
`commitTransfer`	This command is used to report the results of a transfer request.
`getKeyPair`	This command is used in EGM-controlled mode to request a new key for PIN encryption.

Table 6.28 `wat` Class Commands Originated by Host

Command	Description
`setWatState`	This command is used to enable or disable the WAT functionality on an EGM.
`setWatLockOut`	This command is used to request that the device lock the EGM.
`getWatStatus`	This command is used to request the current status of the WAT functionality.
`getWatProfile`	This command is used to request the current WAT configuration.
`setWatCashOut`	This command is used to tell the EGM which WAT device to use for EGM-initiated cash-outs.
`initiateRequest`	This command is used in host-controlled mode to request that the EGM to initiate a transfer.
`cancelRequest`	This command is used in host-controlled mode to cancel a pending transfer.
`getWatLogStatus`	This command is used to request the current status of the class-level log.
`getWatLog`	This command is used to request the contents of the class-level log.

6.8.6 Events

Table 6.29 `wat` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM

Table 6.29 `wat` Class Events

Event
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
WAT Cash-Out to Host Disabled
WAT Cash-Out to Host Enabled
WAT Transfer Requested by Host
WAT Transfer Cancellation Requested by Host
WAT Transfer Initiated
WAT Transfer Authorized
WAT Transfer Completed
WAT Commit Command Acknowledged
New Key Pair Established
New Key Pair Failed - Invalid Key Value
Cash Out To Host Pending
Cash Out To Host Pending Cleared

6.8.7 meters

Table 6.30 `wat` Class Meters

Meters
Value of cashable amount transferred in.
Value of promotional amount transferred in.
Value of non-cashable amount transferred in.
Count of transfers in.
Value of cashable amount transferred out.
Value of promotional amount transferred out.

Table 6.30 `wat` Class Meters

Meters
Value of non-cashable amount transferred out.
Count of transfers out.

6.9 smartCard

Benefit: Smart cards can also trigger player tracking sessions.

The `smartCard` class is used to manage cashless transfers to/from smart cards. The transfers can be performed with or without direct host interaction depending on the smart card application.

Funds can be cashable, promotional, or non-cashable. Not all smart card applications will support all credit types.

The `smartCard` class uses all four G2S building blocks: status, profile, meters, and logs.

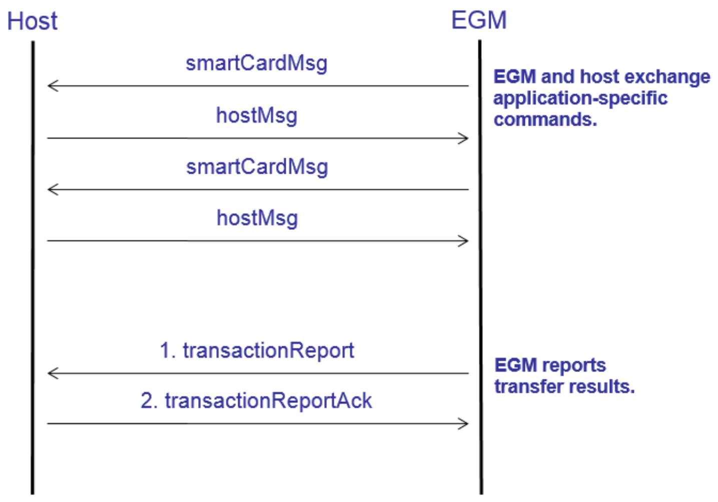
6.9.1 Command Sequence

Benefit: Smart-card transfers can be managed by the EGM through a secure transaction module or via application-specific commands exchanged with a host.

The sequence diagram below illustrates how commands in the `smartCard` class are intended to be used.

If the smart card application requires host interaction, first, the EGM and host exchange `smartCardMsg` and `hostMsg` commands. These commands contain applications-specific instructions related to the smart card transfer.

After the transfer has been made, the EGM sends a `transactionReport` command to the host reporting the results of the transfer. The host acknowledges this with the `transactionReportAck` command. The EGM must retry the `transactionReport` command until acknowledged by the host.



6.9.2 Commands

Table 6.31 `smartCard` Class Commands Originated by EGM

Command	Description
`smartCardMsg`	This command is used to send application-specific instructions to the host.
`transactionReport`	This command is used to report the results of a smart card transaction.

Table 6.32 `smartCard` Class Commands Originated by Host

Command	Description
`setSmartCardState`	This command is used to enable or disable the `smartCard` device functionality.
`setSmartCardLockOut`	This command is used to request that the device lock the EGM.
`getSmartCardStatus`	This command is used to request the current status of the `smartCard` device.
`getSmartCardProfile`	This command is used to request the current configuration of the `smartCard` device.
`getSmartCardLogStatus`	This command is used to request the current status of the class-level transaction log.
`getSmartCardLog`	This command is used to request the contents of the class-level transaction log.
`hostMsg`	This command is used to send application-specific instructions to the EGM.

6.9.3 Events

Table 6.33 `smartCard` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host

Table 6.33 smartCard Class Events

Event
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host.
EGM Lock Not Requested by Host.
Smart Card Validated
Smart Card Removed
Smart Card PIN Locked
Smart Card PIN Verified
Smart Card Blocked
Smart Card Not Blocked
Cash Out to Smart Card Pending
Cash Out to Smart Card Pending Cleared
Transfer Initiated
Transfer Completed
Transfer Failed
Transfer Report Acknowledged
Smart Card Status Changed
Secure Transaction Module Serial Number Changed
Secure Transaction Module Authorized
Secure Transaction Module Not Authorized
Secure Transaction Module Blocked
Secure Transaction Module Not Blocked
Secure Transaction Module Near Write Maximum
Secure Transaction Module Not Near Write Maximum

6.9.4 Meters

Table 6.34 `smartCard` Class Meters

Meter
Value of cashable amount transferred in.
Value of promotional amount transferred in.
Value of non-cashable amount transferred in.
Count of transfers in.
Value of cashable amount transferred out.
Value of promotional amount transferred out.
Value of non-cashable amount transferred out.
Count of transfers out.

6.10 dft (direct funds transfer)

The `dft` class is used to manage the process of transferring funds directly between an EGM and a host system application (rather than to and from a wagering account). For example, to purchase a keno ticket or enter into a tournament.

The `dft` class uses all four G2S building blocks: status, profile, meters, and logs.

6.10.1 Command Sequence

Benefit: Funds can be applied to the credit meter or paid to a ticket.

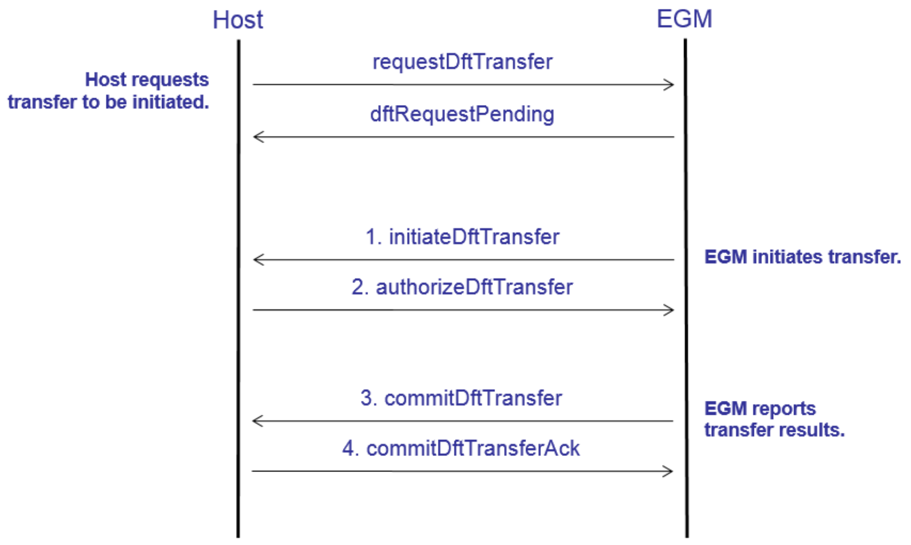
The sequence diagram below illustrates how commands in the `dft` class are intended to be used.

First, the host requests that the EGM start a transfer with the `requestDft-Transfer` command. The EGM responds with the `dftRequestPending` command. The EGM will wait until the end of the game cycle to initiate the transfer.

When ready to initiate the transfer, the EGM uses `initiateDftTransfer` to start the actual transfer.

The host responds with the `authorizeDftTransfer` command.

Once the transfer is complete, the EGM sends a `commitDftTransfer` command to the host. The EGM will retry the command until it is acknowledged by the host.



6.10.2 Commands

Table 6.35 `dft` Class Commands Originated by EGM

Command	Description
`initiateDftTransfer`	This command is used to initiate a previously requested direct funds transfer.
`commitDftTransfer`	This command is used to report the final result of a previously initiated direct funds transfer.

Table 6.36 `dft` Class Commands Originated by Host

Command	Description
`setDftState`	This command is used to enable or disable `dft` device functionality on an EGM.
`setDftLockOut`	This command is used to request that the device lock the EGM.
`getDftStatus`	This command is used to request the current status of the `dft` device from the EGM.
`getDftProfile`	This command is used to request the current configuration of the `dft` device.
`requestDftTranfer`	This command is used to request that the EGM initiate a direct funds transfer.
`cancelDftRequest`	This command is used to cancel a pending transfer.
`getDftLogStatus`	This command is used to request the current status of the class-level log.
`getDftLog`	This command is used to request the contents of the class-level log.

6.10.3 Events

Table 6.37 `dft` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM

Table 6.37 `dft` Class Events

Event
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Transfer Requested by Host
Transfer Cancel Requested by Host
Transfer Initiated
Transfer Authorized
Transfer Successfully Completed
Transfer Failed
Commit Command Acknowledged

6.10.4 Meters

Table 6.38 `dft` Class Meters

Meter
Value of cashable amount transferred in.
Value of promotional amount transferred in.
Value of non-cashable amount transferred in.
Count of transfers in.
Value of cashable amount transferred out.
Value of promotional amount transferred out.
Value of non-cashable amount transferred out.
Count of transfers out.

CHAPTER

7

Advanced Features Classes

7.1 Introduction

Traditionally, much of the advanced functionality available at an EGM – such as player tracking and tournaments – was provided by proprietary system interface boards connected to serial ports on the EGM. With G2S, that functionality was moved into the EGM itself. There is no need for additional hardware.

The `idReader` and `mediaDisplay` classes are the core of the advanced functionality. The `idReader` class allows the EGM to validate player and employee IDs, making the EGM, as well as host systems, aware of the player or employee present at the EGM. The `mediaDisplay` class provides the Player User Interface (PUI). It allows touch-enabled content to be displayed through pop-up windows on the main screen, or other screens, of the EGM. The content can communicate directly to third-party sytems.

The `player`, `informedPlayer`, `employee`, `tournament`, and `printer` classes build off of the core functionality. The `player` class adds player tracking capabilities. The `informedPlayer` class provides a set of tools that can be used as part of a responsible gaming program. The `employee` class can be used to track employee activities at an EGM. The `tournament` class provides a rich set of tools for managing player tournaments. And, the `printer` class gives the host access to the EGM's printer, allowing player-specific documents to be printed at the EGM.

7.2 idReader

Benefit: SMIBs are not needed to track players and employees.

The idReader class is used to manage ID readers installed on an EGM. Various types of IDs can be used including magnetic-stripe cards, proximity cards, smart cards, fingerprint scanners, direct-entry keypads, etc. The idReader class supports self-validation (smart cards) and host-validation (magnetic-stripe cards).

The idReader class is a multiple-device class. Multiple idReader devices may be active in an EGM.

The idReader class uses three of the four G2S building blocks: status, profile, and meters. There are no logs associated with the idReader class.

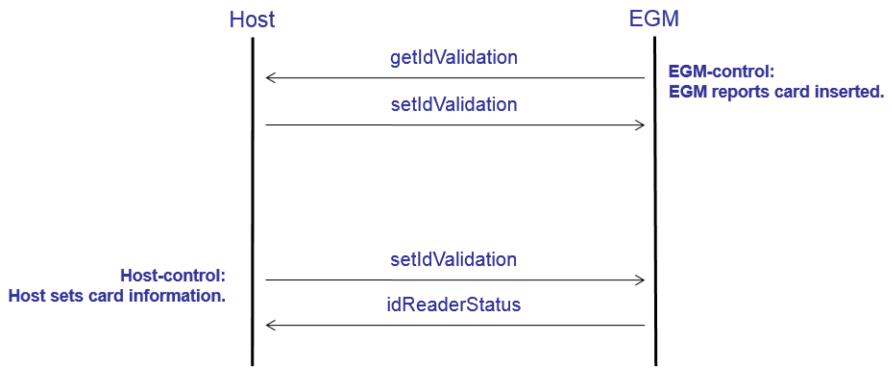
7.2.1 Command Sequence

Benefit: Door access permissions can be set by the host to control reporting of legal/ illegal door opens.

The sequence diagram below illustrates how commands in the idReader class are intended to be used. Two modes of operation are available: EGM-control and host-control.

With EGM-control, the EGM sends the getIdValidation command to the host reporting that a card has been inserted. The host responds with setIdValidation, which contains player or employee information.

With host-control, the host simply sends the setIdValidation command to the EGM, telling the EGM which player or employee is present at the EGM. The EGM responds with the idReaderStatus command.



7.2.2 Commands

Table 7.1 `idReader` Class Commands Originated by EGM

Command	Description
`getIdValidation`	This command is used to request validation of an ID.

Table 7.2 `idReader` Class Commands Originated by Host

Command	Description
`setIdReaderState`	This command is used to enable or disable an `idReader` device.
`getIdReaderStatus`	This command is used to request the current status of an `idReader` device.
`getIdReaderProfile`	This command is used to request the current configuration of an `idReader` device.
`setIdValidation`	This command is used to identify the player or employee associated with an ID.
`getIdReaderLocales`	This command is used to request the current set of text messages used by the EGM for different languages.

7.2.3 Events

Table 7.3 `idReader` Class Events

Events
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
All Device Faults Cleared
Valid ID Presented

Table 7.3 `idReader` Class Events

Events
Invalid ID Presented
ID Cleared From Reader
ID Validated Offline
Unable To Validate Id Offline
Validation Information Changed
ID Validation Expired (ID Abandoned)
Invalid Validation Information Received
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Illegal Activity Detected
Error Detected While Reading ID

7.2.4 Meters

Table 7.4 `idReader` Class Meters

Meter
Number of errors detected while reading IDs.

7.3 player

Benefit: The presence of a valid player ID triggers player tracking: wagers, actual win, theoretical win, and other meter movements.

The `player` class is used to track player activity at an EGM, and based on that activity, provide incentives in the form of points. The `player` class includes commands for reporting player session information, setting point balances and carryover values, and setting point calculation parameters.

Player sessions start when the first game cycle is initiated after a player card is accepted by an ID reader. Player sessions end after the player card is removed and any active game cycles have ended. Player sessions can also be ended if there is no activity at the EGM for a specified period of time.

The `player` class uses all four G2S building blocks: status, profile, meters, and logs.

7.3.1 Command Sequence

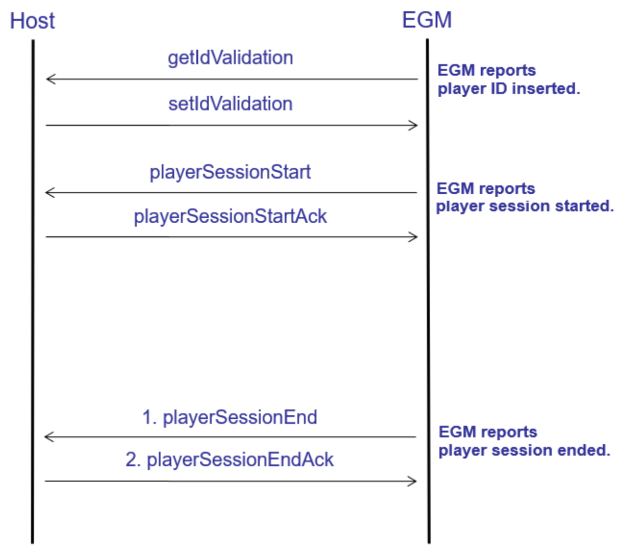
Benefit: Points can be calculated using any of the meters captured during a player session; points can also be awarded by a host.

The sequence diagram below illustrates how commands in the `player` Class are intended to be used.

First, the EGM sends a `getIdValidation` command to the host to validate the player's ID. The host responds with the `setIdValidation` command.

Once the player session is started, the EGM sends a `playerSessionStart` command to the host. The host acknowledges this with the `playerSessionStartAck` command. The player session starts regardless of whether an acknowledgement is received.

When the player session ends, the EGM sends a `playerSessionEnd` command. The host acknowledges this with a `playerSessionEndAck` command. The EGM must retry the `playerSessionEnd` command until it is acknowledged.



7.3.2 Commands

Table 7.5 `player` Class Commands Originated by EGM

Command	Description
`getCountdownOverride`	This command is used to request the current generic override for point calculations from the host.
`playerSessionStart`	This command is used to notify the host when a player session has started.
`playSessionEnd`	This command is used to notify the host when a player session has ended.

Benefit: Three different point calculations can be active — default, player-specific, and generic (time-of-day); the player receives the best award out of the three.

Table 7.6 `player` Class Commands Originated by Host

Command	Description
`setPlayerState`	This command is used to enable or disable the `player` device.
`getPlayerStatus`	This command is used to request the current status of the `player` device.
`getPlayerProfile`	This command is used to request the current configuration of the `player` device.
`setCountdownOverride`	This command is used to set the generic over-ride for point calculations.

Table 7.6 `player` Class Commands Originated by Host

Command	Description
`setCarryOver`	This command is used to set the initial point carryover for a player session.
`setPointBalance`	This command is used to set the player's current point balance.
`setPlayerOverride`	This command is used to set the player-specific override parameters for point calculations.
`setHostPoints`	This command is used to award additional points to the player during a session.
`setPlayerMessage`	This command is used to send an EGM a message to display to a player.
`getPlayerLogStatus`	This command is used to request the current status of the class-level log.
`getPlayerLog`	This command is used to request the contents of the class-level log.
`getPlayerLocales`	This command is used to request the current text messages used for different languages by a `player` device.
`getValidationDevices`	This command is used to request the current list of `idReader` devices associated with the `player` device.

7.3.3 Events

Benefit: Up to five hot player levels can be set; when a player reaches a new level, an event is generated.

Table 7.7 `player` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Session Started
Interval Rating
Carded Hot Player Level Changed

Table 7.7 `player` Class Events

Event
Session Ended
Session End Acknowledged
RPN Processing Fault
Generic Countdown Override Set
Generic Countdown Override Activated
Player Override Set
Player Override Activated
Base Countdown Activated
Player Carryover Set
Player Point Balance Set
Host Points Set
Session Updated
Un-Carded Hot Player Level 0
Un-Carded Hot Player Level 1
Un-Carded Hot Player Level 2
Un-Carded Hot Player Level 3
Un-Carded Hot Player Level 4
Un-Carded Hot Player Level 5
Point Limit Exceeded
Too Many Points Earned - No Award
Player Session Failed to Start

7.3.4 Meters

Table 7.8 `player` Class Meters

Meter
Total cashable credits wagered on the EGM while a player session was active.
Total promotional credits wagered on the EGM while a player session was active.

Table 7.8 `player` Class Meters

Meter
Total non-cashable credits wagered on the EGM while a player session was active.
Total base paytable win while a player session was active.
Total progressive win while a player session was active.
Total external bonus awards transferred as win while a player session was active.
Number of games played while player session was active.

7.4 informedPlayer

Benefit: Features in this class can be used to implement a responsible gaming program.

The `informedPlayer` class provides a set of tools that can be used as part of a responsible gaming program. It is not a responsible gaming program itself. The `informedPlayer` class includes commands for monitoring player session information, validating player PIN information, and interceding with game play. This can include disabling money in and/or game play, setting the minimum length of a game cycle, disabling continuous play and setting the maximum wager amount.

The `informedPlayer` class uses two of the four G2S building blocks: status and profile. There are no meters in the `informedPlayer` class. The class-level logs from the `player` class are shared with the `informedPlayer` class.

7.4.1 Commands

Benefit: PINs can be used to authenticate players.

Table 7.9 `informedPlayer` Class Commands Originated by EGM

Command	Description
`getIpPinInfo`	This command is used to request PIN authentication information for a player.
`validateIpPin`	This command is used to request that a host validate PIN information for a player.
`ipStatus`	This command is used to report the current status of the `informedPlayer` device.
`getIpKeyPair`	This command is used to request a new security key for PIN encryption from the host.

Table 7.10 `informedPlayer` Class Commands Originated by Host

Command	Description
`setIpState`	This command is used to enable or disable the `informedPlayer` device.
`getIpStatus`	This command is used to request the current status of the `informedPlayer` device.
`getIpProfile`	This command is used to request the current configuration of the `informedPlayer` device.

Table 7.10 `informedPlayer` Class Commands Originated by Host

Command	Description
`setIpMessage`	This command is used to send the EGM a message to display to the player.
`setIpGameDelay`	This command is used to request the EGM initiate a game delay period.

7.4.2 Events

Table 7.11 `informedPlayer` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Money-In Disabled
Money-In Enabled
Game-Play Disabled
Game-Play Enabled
Session Limit Changed
Security Key Changed
Informed Player Game Delay Started
Informed Player Game Delay Ended
Informed Player Host Communications Lost
Informed Player Host Communications Restored
Informed Player Session Start
Informed Player Session Update
Informed Player Session End
Maximum Note Denomination Changed
Maximum Wager Amount Changed

Table 7.11 `informedPlayer` Class Events

Event
Continuous Play Allowed
Continuous Play Not Allowed
Minimum Game-Cycle Changed

7.4.3 Meters

There are no meters associated with the `informedPlayer` class.

7.5 employee

Benefit: The presence of a valid employee ID triggers employee tracking: all meter movements while the employee ID is present.

The `employee` class is used to track employee activity at an EGM. The `employee` class includes commands for reporting employee session information and activity codes. Activity codes can be used to automate machine entry and access logs. An employee can report the reason for entering an EGM, problems discovered, problems resolved, player complaints, etc. Player sessions start and end between game cycles. Employee sessions start and end immediately.

The `employee` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the `employee` class.

7.5.1 Commands

Benefit: Employees can report activity codes indicating why they visited the EGM.

Table 7.12 `employee` Class Commands Originated by EGM

Command	Description
`employeeSessionStart`	This command is used to notify the host that an employee session has started.
`employeeSessionEnd`	This command is used to report that an employee session has ended.
`getEmployeeActivities`	This command is used to request the list of activity codes that an employee is permitted to use.
`employeeActivity`	This command is used to report an employee activity code to the host.
`getEmployeeKeyPair`	This command is used to request a new security key for PIN encryption from the host.
`getEmployeeDevices`	This command is used to request the list of devices that the EGM should use to authenticate employees.
`validateEmployeePIN`	This command is used to request that the host validate a PIN for an employee.

Table 7.13 `employee` Class Commands Originated by Host

Command	Description
`setEmployeeState`	This command is used to enable or disable the employee device.
`getEmployeeStatus`	This command is used to request the current status of the `employee` device.
`getEmployeeProfile`	This command is used to request the current configuration for the `employee` device.
`getEmployeeLogStatus`	This command is used to request the current status of the `employee` session log.
`getEmployeeLog`	This command is used to request the contents of the `employee` session log.
`getActivityLogStatus`	This command is used to request the current status of the activity code log.
`getActivityLog`	This command is used to request the contents of the activity code log.

7.5.2 Events

Table 7.14 `employee` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Employee Session Started
Employee Session Updated
Employee Session Ended
Employee Session Acknowledged
Employee Session Failed To Start
Employee Activity Reported
Employee Activity Acknowledged

Table 7.14 `employee` Class Events

Event
New Key Pair Established
New Key Pair Failed - Invalid Key Value

7.5.3 Meters

There are no meters associated with the `employee` class.

7.6 tournament

Benefit: Tournaments can be configured and activated remotely.

The `tournament` class is used to manage player tournaments and report tournament results. The `tournament` class includes commands for entering players in tournaments, activating tournament sessions, broadcasting tournament standings, closing tournament sessions and reporting tournament session results. The length of a tournament session can be based on the time played, the number of games played, or the number of credits wagered.

The `tournament` class uses all four G2S building blocks: status, profile, meters, and logs.

7.6.1 Command Sequence

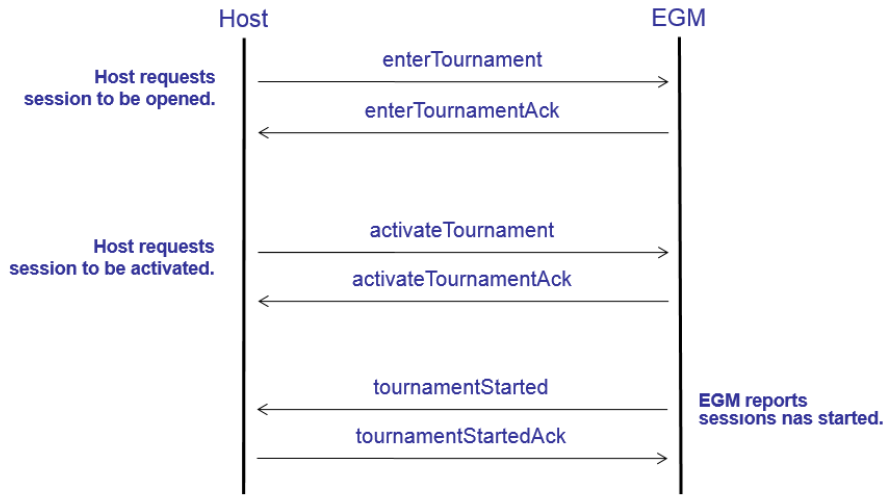
Benefit: Registration fees can be collected via DFT.

The sequence diagram below illustrates how commands in the `tournament` class are intended to be used when a tournament starts.

First the host sends an `enterTournament` command to the EGM. The EGM responds with an `enterTournamentAck` command indicating that the requested tournament session has been opened. Next, when all players are ready, the host sends an `activateTournament` command to the EGM. This starts tournament play.

The EGM responds with an `activateTournamentAck` command. An optional countdown can be used to alert the players that the session is about to begin.

Once any countdowns are finished and, optionally, the first game has started, the EGM sends a `tournamentStarted` command to the host. This command simply indicates that the tournament session has started. The host responds with a `tournamentStartedAck` command.



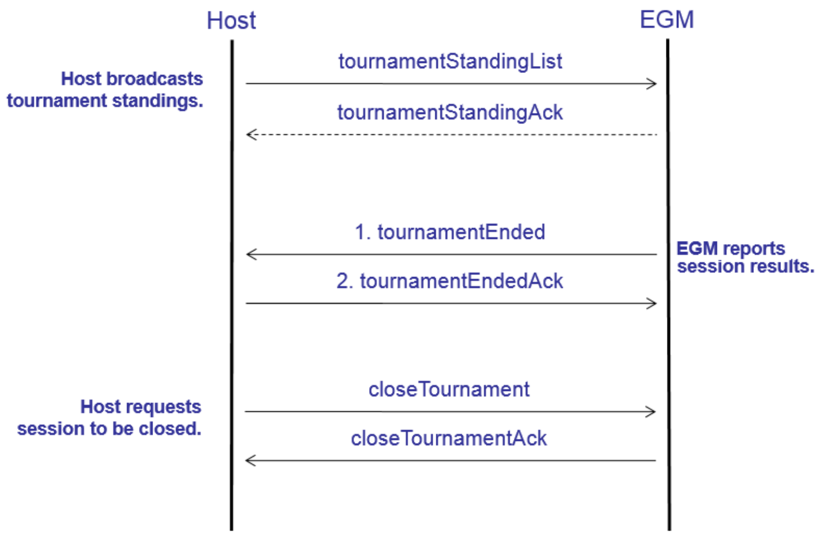
Benefit: Winnings can be paid via DFT.

The next sequence diagram illustrates how commands in the `tournament` class are intended to be used when a tournament ends.

First, during tournament play the host can use the `tournamentStandingList` command to broadcast current tournament standings to the EGMs. The `tournamentStandingList` command can be sent point-to-point or via multicast. When sent point-to-point, a `tournamentStandingAck` command is sent in response.

When the tournament session ends, the EGM sends a `tournamentEnded` command to the host containing the final results of the session. The hosts returns a `tournamentEndedAck` command to acknowledge receipt of the results. The `tournamentEnded` command must be retried until acknowledged.

Once all results from all EGMs are collected and winners are determined, the host sends a `closeTournament` command to the EGM. This allows the EGM to enter another tournament or return to regular game play. The EGM responds with a `closeTournamentAck` command.



7.6.2 Commands

Table 7.15 `tournament` Class Commands Originated by EGM

Command	Description
`getTournaments`	This command is used to request the list of tournaments that are available to the player.
`registerTournament`	This command is used to request that the host enter a player into a tournament.
`tournamentStarted`	This command is used to report that a tournament session has been started.
`tournamentEnded`	This command is used to report that a tournament session has ended.
`getPlayerList`	This command is used to obtain the list of players in the specified tournament.

Table 7.16 `tournament` Class Commands Originated by Host

Command	Description
`setTournamentState`	This command is used to enable or disable the `tournament` device.
`enterTournamentMode`	This command is used to request than the EGM enter tournament mode.

Table 7.16 `tournament` Class Commands Originated by Host

Command	Description
`getTournamentStatus`	This command is used to request the current status of the `tournament` device.
`getTournamentProfile`	This command is used to request the current configuration of the `tournament` device.
`getTournamentGames`	This command is used to request a list of games that can be used for tournament play.
`enterTournament`	This command is used to instruct the EGM to enter a tournament.
`activateTournament`	This command is used to activate a tournament session.
`cancelTournament`	This command is used to cancel a tournament session.
`closeTournament`	This command is used to notify an EGM that a tournament session can be closed - the EGM can return to normal game play.
`tournamentStandingList`	This command is used to broadcast the current standings for the tournament.
`getTournamentLogStatus`	This command is used to request the current status of the class-level log.
`getTournamentLog`	This command is used to request the contents of the class-level log.
`tournamentSync`	This command is used to synchronize the countdown, count-out, and session time remaining in a tournament.
`playerList`	This command is used to send the list of player names to an EGM.
`tournamentSummaryList`	This command is used to broadcast a summarized list of the current standings for a tournament to an EGM.

7.6.3 Events

Table 7.17 `tournament` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM

Table 7.17 `tournament` Class Events

Event
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
Tournament Mode Enabled
Tournament Mode Not Enabled
Tournament Entry Requested
Tournament Entry Voided
Tournament Session Pending
Player Left Tournament Session
Host Cancelled Tournament Session
Tournament Session Activated
Tournament Session Started
Tournament Session Updated
Player Abandoned Tournament
Game Play Fault
Tournament Session Completed
Tournament Session Closed
Player Name Changed
Tournament Count-Out Started

7.6.4 Meters

Table 7.18 `tournament` Class Meters

Meter
Total number of tournament games played.
Total number of tournament credits wagered.
Total number of tournament credits won.

7.7 printer

Benefit: Hosts can define printer templates and then use those templates to print documents – receipts, coupons, promotional offers, etc.

The `printer` class is used to configure and print documents from an EGM.

The class is designed to be used with printers that support the *GDS Printer Description Language (PDL)*. Templates are used to define standard printed documents.

The `printer` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the `printer` class.

7.7.1 Commands

Table 7.19 `printer` Class Commands Originated by EGM

Command	Description
	None.

Table 7.20 `printer` Class Commands Originated by Host

Command	Description
`getPrinterProfile`	This command is used to request the current configuration of a `printer` device.
`setPrinterState`	This command is used to enable or disable a `printer` device.
`getPrinterStatus`	This command is used to request the current status of a `printer` device.
`getPrinterTemplates`	This command is used to request the current list of templates supported by the `printer` device.
`printTicket`	This command is used to send a request that the EGM print a document.
`getPrintLogStatus`	This command is used to request the current status of the class-level log.
`getPrintLog`	This command is used to request the contents of the class-level log.

7.7.2 Events

Table 7.21 `printer` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host
Device Configuration Changed by Host
Device Configuration Changed by Operator
All Device Faults Cleared
Printer Transfer Failed
Print Completed
Print Failed
Printer Chassis Opened
Printer Chassis Closed
Print Head Opened
PrintHead Closed
Paper Jam
Paper Low
Paper Empty
Device Disconnected
Device Connected
Device Firmware Fault
Device Mechanical Fault
Device Optical Fault
Device Component Fault
Device Non-Volatile Memory Fault
Paper Not At Top Of Form
Paper Not Low
Illegal Activity Detected

7.7.3 Meters

There are no meters associated with the `printer` class.

7.8 mediaDisplay

Benefit: Content written in HTML or Flash can be loaded, activated, and displayed to players.

The `mediaDisplay` class is used to manage *Player User Interface* (PUI) windows and the display of third-party content on EGMs. There can be multiple media displays displayed on a given screen at a given time.

The class includes commands for loading content onto an EGM, activating the content in a window, and then displaying the window.

A wide variety of applications can be run in PUI windows. Player tracking, wagering accounts, and sports betting applications are just some of the possibilities. Multiple independent windows can be configured into the EGM. The content can be written in HTML 5 or Flash 7.

The `mediaDisplay` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the `mediaDisplay` class.

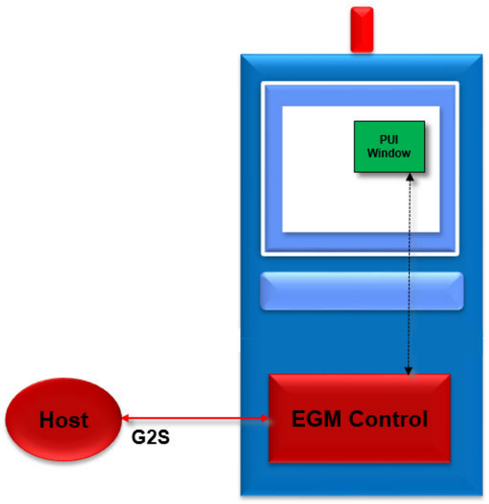
7.8.1 mediaDisplay Configuration

Windows are defined in the profile of the EGM. The profile identifies the screen on which the windows should be displayed (main screen, top box, etc), the position (top, bottom, left, or right), the size (x and y dimensions), and the content types that are supported (Flash 7, HTML 5).

Each window is a separate `mediaDisplay` device. The windows can be configured independently of each other and different host systems can control different windows.

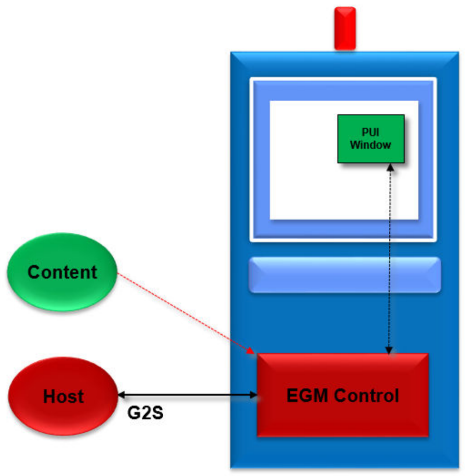
7.8.2 mediaDisplay Management

The host system manages the window via the G2S protocol. The host is able to load and activate content within a PUI window. The host can also display and hide the window. Finally, the host may also release content.



7.8.3 mediaDisplay Content

The EGM loads content from the location specified by the host. Content can be pre-loaded for fast transitions between different content. All content is released (deleted) when the EGM restarts.



7.8.4 mediaDisplay Applications

Benefit: PUI content can communicate directly with the EGM through the EMDI protocol.

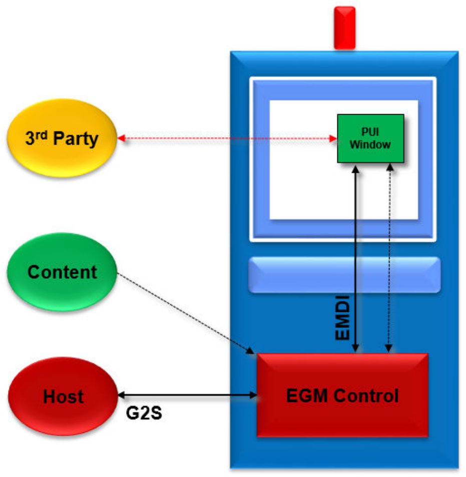
Content running in the window can communicate directly to the EGM via the EGM Media Display Interface (EMDI).

Benefit: PUI content can communicate with third-party application servers.

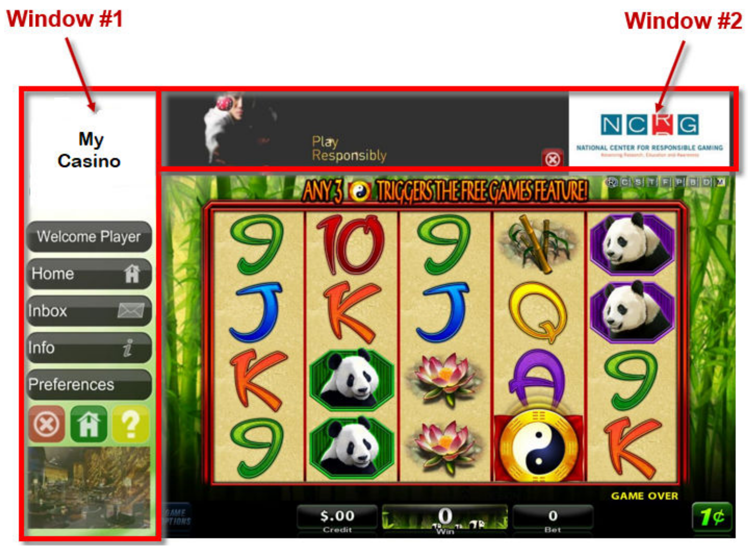
The content can react to events as they happen on the EGM. The content can also request that the EGM take certain actions related to the window – hide, show, etc.

Benefit: Third-party applications can add/remove funds from the EGM via DFT.

Content can also communicate directly to third-party application systems, such as promotions, hotel, restaurant, sports books, and tournaments.



The following illustration contains an example of a game with two PUI windows. Window 1 contains a player service window offering various third-party applications to the player. Window 2 contains a banner ad.



7.8.5 Commands

Table 7.22 `mediaDisplay` Class Commands Originated by EGM

Command	Description
`contentToHostMessage`	This command is used to send data from PUI content to the host.

Table 7.23 `mediaDisplay` Class Commands Originated by Host

Command	Description
`getMediaDisplayProfile`	This command is used to request the current configuration of the `mediaDisplay` device.
`setMediaDisplayState`	This command is used to enable or disable the functionality of the `mediaDisplay` device.
`getMediaDisplayStatus`	This command is used to request the current status for the `mediaDisplay` device.
`setMediaDisplayLockOut`	This command is used to request that the device lock the EGM.
`loadContent`	This command is used to load content into the `mediaDisplay` device.
`releaseContent`	This command is used to release content from the `mediaDisplay` device.
`setActiveContent`	This command is used to specify which content should be active in the `mediaDisplay` device.
`getContentStatus`	This command is used to request the current status of content loaded in a `mediaDisplay` device.
`showMediaDisplay`	This command is used to make a `mediaDisplay` device visible.
`hideMediaDisplay`	This command is used to hide a `mediaDisplay` device.
`getContentLogStatus`	This command is used to request the current status of the class-level log.
`getContentLog`	This command is used to request the contents of the class-level log.

Table 7.23 `mediaDisplay` Class Commands Originated by Host

Command	Description
`hostToContentMessage`	This command is used to send data from the host to PUI content.
`raiseMediaDisplay`	This command is used to cause an overlay window to become the topmost window displayed to the user.

7.8.6 Events

Table 7.24 `mediaDisplay` Class Events

Event
Media Display Disabled By EGM
Media Display Enabled By EGM
Media Display Disabled By Host
Media Display Enabled by Host
Media Display Configuration Changed by Host
Media Display Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Content Pending
Content Loaded
Content Executing
Content Released
Content Error
Media Display Shown
Media Display Hidden
EGM Local Media Display Interface Open
EGM Local Media Display Interface Closed
EGM Local Media Display Interface Failure
Window Topmost
Window Not Topmost
Window Modal

Table 7.24 `mediaDisplay` Class Events

Event
Window Not Modal
Game Play Suspended
Game Play Not Suspended

7.8.7 Meters

There are no meters associated with the `mediaDisplay` class.

CHAPTER

8

Configuration & Download Classes

8.1 Introduction

Remote configuration and download are two truly unique features of G2S. They allow operators to manage the software and configuration of their EGMs from a central location, helping to improve the efficiency and accuracy of EGM management.

The `optionConfig` class allows the operational parameters of an EGM to be set remotely by a host system. The `commConfig` class is used to manage the G2S communications parameters.

The `download` class provides software download and installation services, allowing game software and peripheral firmware to be updated remotely. The `storage` and `hardware` classes are used to verify software compatibility prior to download and installation. The `gat` class gives operators and regulators the opportunity to authenticate the software and firmware on an EGM, assuring that the download and installation process was successful and that the EGM is in compliance with jurisdictional requirements.

8.2 optionConfig

Benefit: Protocol-related options and manufacturer-defined options can both be configured remotely.

The optionConfig class is used to set the operational parameters of the EGM (the profile attributes). Both protocol-related and manufacturer-defined parameters are supported. The class includes commands for determining the set of configurable options for a device and for requesting changes to those options.

The information reported about the configurable options fully describes the options — title, description, data type, etc. The information can be used to dynamically build a user interface.

The optionConfig class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the optionConfig class.

8.2.1 Command Sequence

The sequence diagram below illustrates how commands in the optionConfig class are intended to be used. It is the same sequence that should be followed for the commConfig and download classes.

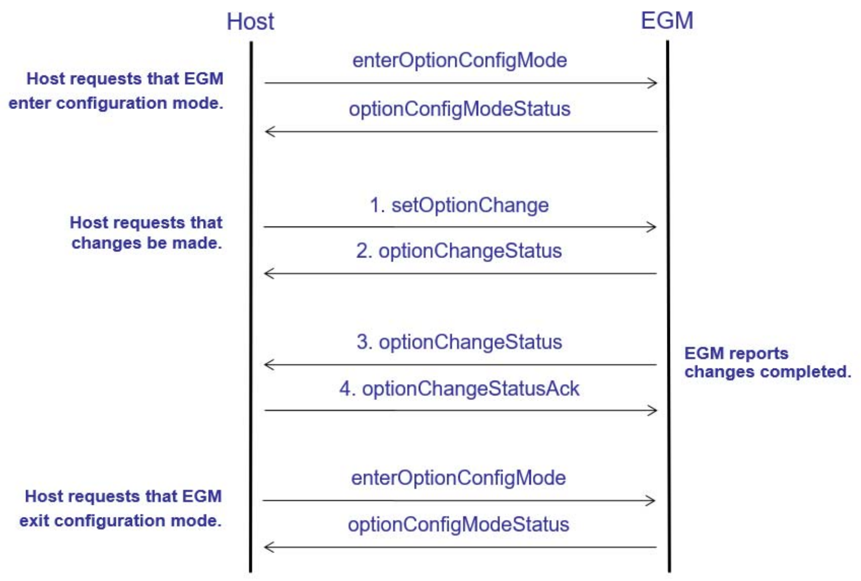
First, the host sends a enterOptionConfigMode command to the EGM requesting that the EGM disable itself and enter configuration mode. Once idle requirements have been satisfied, the EGM will respond with the optionConfigModeStatus command.

Next, the host sends a setOptionChange command containing a set of configuration changes. The EGM responds with the optionChangeStatus command indicating that the change request is in process.

After changes have been applied, the EGM sends an optionChangeStatus command indicating that the change request has been completed. The host responds with the optionChangeStatusAck command.

At this point, the host could request other changes — optionConfig, commConfig, or download changes.

Lastly, the host sends a enterOptionConfigMode command requesting that the EGM exit configuration mode. The same command is used to enter and exit configuration mode. Once the post-configuration idle requirements have been met, the EGM responds with a optionConfigModeStatus command indicating that the EGM is no longer in configuration mode.



8.2.2 Commands

Table 8.1 `optionConfig` Class Commands Originated by EGM

Command	Description
`optionList`	This command is used to send a list of device options to a host.
`optionChangeStatus`	This command is used to report the status of a set of option changes.

Table 8.2 `optionConfig` Class Commands Originated by Host

Command	Description
`enterOptionConfigMode`	This command is used to enable or disable the configuration mode.
`getOptionConfigModeStatus`	This command is used to request the current status of the `optionConfig` device.
`getOptionConfigProfile`	This command is used to request the configuration information for the `optionConfig` device.

Table 8.2 `optionConfig` Class Commands Originated by Host

Command	Description
`getOptionList`	This command is used to request the list of options and current settings from an EGM.
`setOptionChange`	This command is used to send a set of option changes to an EGM.
`cancelOptionChange`	This command is used to cancel a set of option changes.
`authorizeOptionChange`	This command is used to authorize an EGM to apply a set of changes.
`getOptionChangeLogStatus`	This command is used to request the current status of the class-level log.
`getOptionChangeLog`	This command is used to request the contents of the class-level log.
`getOptionSeries`	This command is used to request a subset of the options and current settings from an EGM.

Table 8.3 `optionConfig` Class Events

Event
Device Configuration Changed by Host
Device Configuration Changed by Operator
Entered Configuration Mode
Exited Configuration Mode
Configuration Change Pending
Configuration Change Authorized by Host
Configuration Change Cancelled
Configuration Change Applied
Configuration Change Aborted
Configuration Change Error
Waiting on Authorization
Authorization Timeout

8.2.3 Meters

There are no meters associated with the `optionConfig` class.

8.3 commConfig

Benefit: Communications between EGMs and hosts can be configured remotely.

The `commConfig` class allows a host system to change the G2S communications parameters of the EGM — that is, setting the list of registered hosts and assigning owners, configurators, and guests to devices.

The `commconfig` class uses three of the four G2S building blocks: status, profile, and logs. There are no meters in the `commconfig` class.

8.3.1 command Sequence

The command sequence used for `optionConfig` changes is also used for `commConfig` changes. See Section *8.2, optionConfig* for more details.

8.3.2 Commands

Table 8.4 `commConfig` Class Commands Originated by EGM

Command	Description
`commHostList`	This command is used to report the current communications configuration structure for the EGM.
`commChangeStatus`	This command is used to report the status of a set of changes.

Table 8.5 `commConfig` Class Commands Originated by Host

Command	Description
`enterCommConfigMode`	This command is used to enable or disable the configuration mode.
`getCommConfigModeStatus`	This command is used to request the current status of the `commConfig` device.
`getCommConfigProfile`	This command is used to request the configuration information for the `commConfig` device.
`getCommHostList`	This command is used to request the G2S communications parameters from the EGM.
`setCommChange`	This command is used to send a set of changes to the G2S communication parameters to an EGM.

Table 8.5 commConfig Class Commands Originated by Host

Command	Description
`cancelCommChange`	This command is used to cancel a set of changes.
`authorizeCommChange`	This command is used to authorize an EGM to apply a set of changes.
`getCommChangeLogStatus`	This command is used to request the current status of the class-level log.
`getCommChangeLog`	This command is used to request the contents of the class-level log.

8.3.3 Events

Table 8.6 commConfig Class Events

Event
Device Configuration Changed by Host
Device Configuration Changed by Operator
Entered Configuration Mode
Exited Configuration Mode
Configuration Change Pending
Configuration Change Authorized by Host
Configuration Change Cancelled
Configuration Change Applied
Configuration Change Aborted
Configuration Change Error
Waiting on Authorization
Authorization Timeout

8.3.4 Meters

There are no meters associated with the `commConfig` class.

8.4 download

Benefit: Software packages can be downloaded and installed onto EGMs and their peripheral devices.

The `download` class allows a host system to manage the download, upload, installation, and removal of software packages related to an EGM.

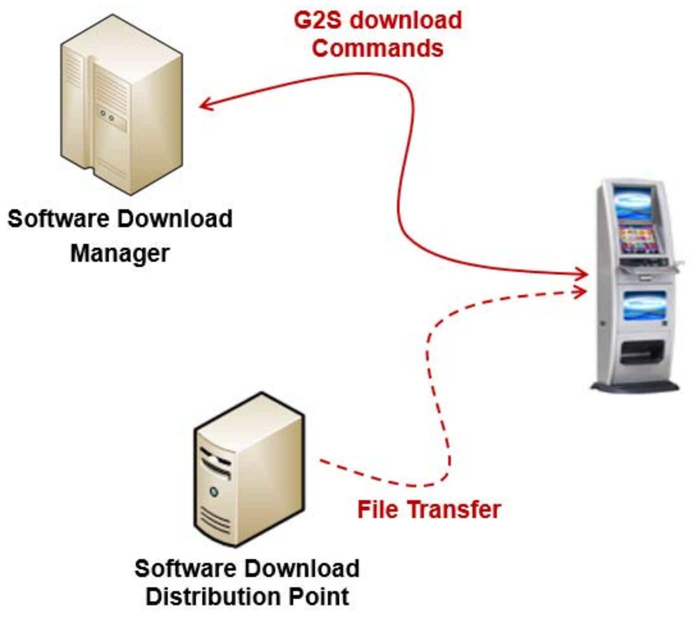
The `download` class uses three of the standard building blocks of G2S: status, profile, and logs. There are no meters associated with the `download` class.

The illustration below illustrates the download concept.

On the top is a Software Download Manager (SDM). The SDM uses download commands to manage the download and installation process.

On the bottom is a Software Download Distribution Point (SDDP). It contains the actual software packages. The SDDP can be local, in the casino, or remote.

When the EGM receives a command from the SDM to download a package, the EGM uses a file transfer protocol, such as ftp or http, to retrieve the package from the SDDP.


G2S download Commands
Software Download Manager
File Transfer
Software Download Distribution Point

Scripts are used to manage the installation of packages. Scripts may contain one or more commands.

Three types of commands are available: package commands, module commands and system commands. For example, install package A, install package B, uninstall module C and then restart the EGM.

8.4.1 Command Sequence

The sequence diagram below illustrates how commands in the `download` class are intended to be used when installing a package. It is similar to the sequence diagram shown for the G2S configuration classes.

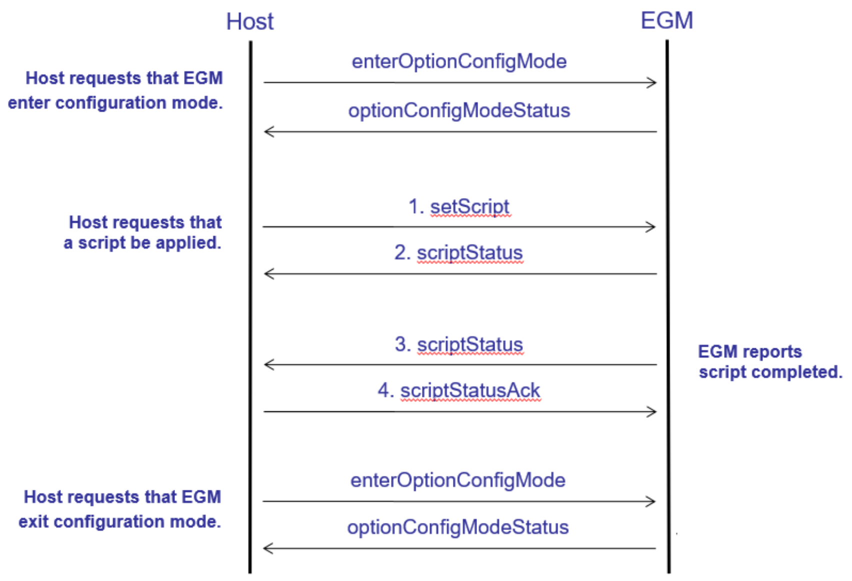
First, the host puts the EGM into configuration mode. The host does this by sending an `enterOptionConfigMode` (or `enterCommConfigMode`) command to the EGM requesting that the EGM disable itself and enter configuration mode.

Next, the host applies the script. The host sends a `setScript` command to the EGM containing the package installation instructions.

After the script has been run, the EGM sends a `scriptStatus` command indicating that the script was completed. The host responds with a `scriptSta-tusAck` command.

At this point, the host can request that other scripts be run or that other changes be made — `optionConfig` or `commConfig` changes.

Lastly, the host sends an `enterOptionConfigMode` command requesting that the EGM exit configuration mode. The same command is used to enter and exit configuration mode.


Host
EGM
Host requests that EGM enter configuration mode.
enterOptionConfigMode
optionConfigModeStatus
Host requests that a script be applied.
1. setScript
2. scriptStatus
3. scriptStatus
4. scriptStatusAck
EGM reports script completed.
Host requests that EGM exit configuration mode.
enterOptionConfigMode
optionConfigModeStatus

8.4.2 Commands

Table 8.7 download Class Commands Originated by EGM

Command	Description
packageStatus	This command is used to send the status of a package to a host.
scriptStatus	This command is used to send the status of a script to a host.

Table 8.8 download Class Commands Originated by Host

Command	Description
setDownloadState	This command is used to enable or disable the functionality of the download device.
getDownloadStatus	This command is used to request the status of the download device.
getDownloadProfile	This command is used to request the configuration of the download device.
addPackage	This command is used to direct the EGM to download a package and add it to the packageList.
createPackage	This command is used to direct the EGM to create a package and add it to the packageList.
deletePackage	This command is used to direct the EGM to delete a package from the packageList.
uploadPackage	This command is used to direct the EGM to upload a package to a specified location.
getPackageStatus	This command is used to request the status of a package.
readPackageContents	This command is used to request a list of the contents of a package.
getPackageList	This command is used to request the list of packages available on the EGM.
getPackageLogStatus	This command is used to request the status of the package log.
getPackageLog	This command is used to request the contents of the package log.

Table 8.8 `download` Class Commands Originated by Host

Command	Description
`setScript`	This command is used to request that the EGM execute a script.
`cancelScript`	This command is used to request that the EGM cancel a script.
`getScriptStatus`	This command is used to request the status of a script.
`authorizeScript`	This command is used to authorize the initiation of a script.
`getScriptList`	This command is used to request the list of pending scripts on the EGM.
`getScriptLogStatus`	This command is used to request the status of the script log.
`getScriptLog`	This command is used to request the contents of the script log.
`getModuleList`	This command is used to request the modules installed on the EGM.
`getSupportedProtocols`	This command is used to request the list of transfer protocols supported by an EGM.
`abortPackageTransfer`	This command is used to direct the EGM to abort a specified package download or upload.
`pausePackageTransfer`	This command is used to direct the EGM to pause the specified download or upload operation.
`resumePackageTransfer`	This command is used to direct the EGM to resume the specified download or upload operation.

8.4.3 Events

Table 8.9 `download` Class Events

Event
Device Disabled by EGM
Device Not Disabled by EGM
Device Disabled by Host
Device Not Disabled by Host

Table 8.9 `download` Class Events

Event
Device Configuration Changed by Host
Device Configuration Changed by Operator
EGM Lock Requested by Host
EGM Lock Not Requested by Host
Package Added
Package Download Initiated
Package Download Refused
Insufficient Storage for Package Download
Package Download in Progress
Package Download Complete
Package Download Aborted
Package Passed Validation
Package Failed Validation
Package Created
Package Upload Requested
Package Upload Initiated
Package Upload Refused
Package Upload in Progress
Package Upload Complete
Package Upload Aborted
Package Deleted
Script Set
Script Authorization Enabled by Host
Script Authorization Disabled by Host
Script Time Started
Script Time Ended
Script EGM Disable Started
Script Waiting on Authorization
Script Waiting on Operator Initiation
Script Initiated, Commands Executing
Script Completed

Table 8.9 `download` Class Events

Event
Script Error
Script Cancelled
Module Added
Module Deleted
Module Enabled
Module Disabled
Module Error
Package Transfer Paused
Script Waiting on Start Date/Time
Script Waiting on EGM Disable

8.4.4 Meters

There are no meters associated with the `download` class.

8.5 storage

The `storage` class allows a host to determine whether an EGM has sufficient storage space available to download and install packages. The available storage can be compared to the prerequisites for the packages to determine whether a proposed operation is feasible.

8.5.1 Commands

Table 8.10 `storage` Class Commands Originated by EGM

Command	Description
	None.

Table 8.11 `storage` Class Commands Originated by Host

Command	Description
`getStorageInfo`	This command is used to request the storage information from the EGM.

8.5.2 Events

There are no events associated with the `storage` class.

8.5.3 Meters

There are no meters associated with the `storage` class.

8.6 hardware

The `hardware` class allows a host to request the list of hardware devices installed on an EGM. The list includes the devices supported by IGSA's GDS standards (coin acceptors, note acceptors, hoppers, etc.) as well as other devices, such as monitors and button panels.

The hardware information can be compared to the prerequisites for the packages to determine whether a proposed operation is feasible.

8.6.1 Commands

Table 8.12 `hardware` Class Commands Originated by EGM

Command	Description
	None.

Table 8.13 `hardware` Class Commands Originated by Host

Command	Description
`getHardwareStatus`	This command is used to request the status of a `hardware` device.
`getHardwareDevices`	This command is used to request the list of hardware devices present on an EGM.

8.6.2 Events

Table 8.14 `hardware` Class Events

Event
Hardware Device List Updated

8.6.3 Meters

There are no meters associated with the `hardware` class.

8.7 gat

Benefit: Software installed on the EGM and peripheral devices can be verified remotely.

The `gat` class allows a host to request verification of software installed on an EGM. This is especially important after new software has been installed on an EGM through the `download` class.

Various algorithms can be used to verify the software and firmware. HMAC SHA-1 is preferred for EGMs; CRC-32 is preferred for peripheral devices.

GAT (Game Authentication Terminal) was originally published as a serial-based protocol to meet regulatory requirements related to the alterable media. As gaming has evolved from serial-based protocols to network-based protocols, GAT has evolved too. The G2S `gat` class makes the same functionality that was available through the original GAT protocol also available through G2S. The functionality is the same; the access methods are different.

The `gat` class uses two of the G2S standard building blocks: profile and logs. There are no status or meters associated with the `gat` class.

8.7.1 Command Sequence

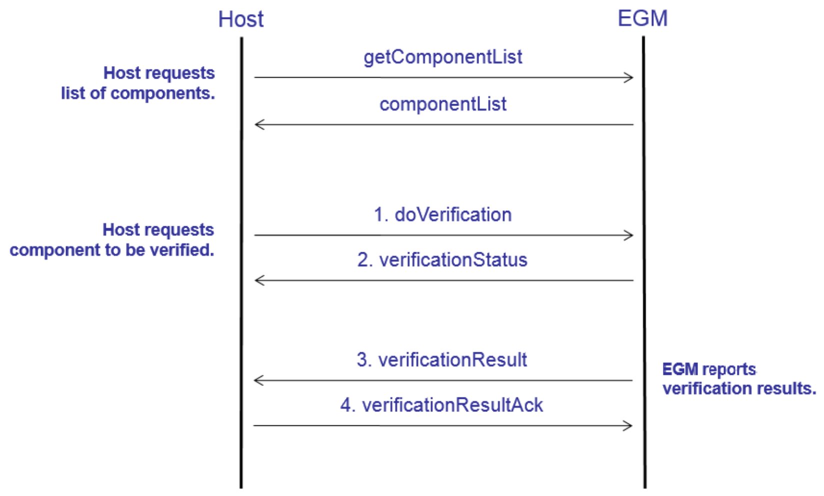
The sequence diagram below illustrates how commands in the `gat` class are intended to be used.

First, the host requests the list of components installed on the EGM using the `getComponentList` command. The EGM responds with a `componentList` command. The `componentList` command identifies the components installed on the EGM as well as the algorithms that can be used to authenticate them.

Next, the host requests that a component be verified with the `doVerification` command. The EGM responds with a `verificationStatus` command indicating that the request is pending.

Multiple requests can be pending on an EGM. The EGM can choose to do the calculations one-at-a-time or in-parallel. Typically, the calculations are performed by a low-priority background process.

After the calculation has completed, the EGM sends a `verificationResult` command to the host containing the software signature. The host responds with a `verificationResultAck` command. The EGM must retry the `verificationResult` command until it is acknowledged.



8.7.2 Commands

Table 8.15 `gat` Class Commands Originated by EGM

Command	Description
`verificationResult`	This command is used to report the result of a verification request.

Table 8.16 `gat` Class Commands Originated by Host

Command	Description
`getGatProfile`	This command is used to request the current configuration of the `gat` device.
`getComponentList`	This command is used to request the list of components that can be verified.
`doVerification`	This command is used to request verification of a component.
`getVerificationStatus`	This command is used to request the status of a verification request.
`getSpecialFunctions`	This command is used to request the list of special authentication functions supported by the EGM.
`runSpecialFunction`	This command is used to request that an EGM execute a special function.

Table 8.16 `gat` Class Commands Originated by Host

Command	Description
`getGatLogStatus`	This command is used to request the status of the class-level logs.
`getGatLog`	This command is used to request the contents of a class-level log.
`cancelVerification`	This command is used to terminate a `doVerification` request previously submitted by the host.

8.7.3 Events

Table 8.17 gat Class Events

Events
Device Configuration Changed by Host
Device Configuration Changed by Operator
Verification Queued
Verification Started
Verification Complete
Verification Error
Verification Result Acknowledged and Passed
Verification Result Acknowledged and Failed
Special Function Executed
Verification Cancelled

8.7.4 Meters

There are no meters associated with the `gat` class.

Appendix A

This Appendix contains more detailed descriptions of three central G2S concepts:

- G2S Machine Model
- G2S Connection Model
- G2S Message Handling

A.1 G2S Machine Model

The G2S machine model is a "virtual" or "normalized" view of the EGM. It is the view of the EGM as seen through the G2S interface. It is the model that the host interacts with. The "real" EGM is mapped to this model by the manufacturer. The entire machine model must be persisted by the EGM. Understanding the model is essential to working with G2S.

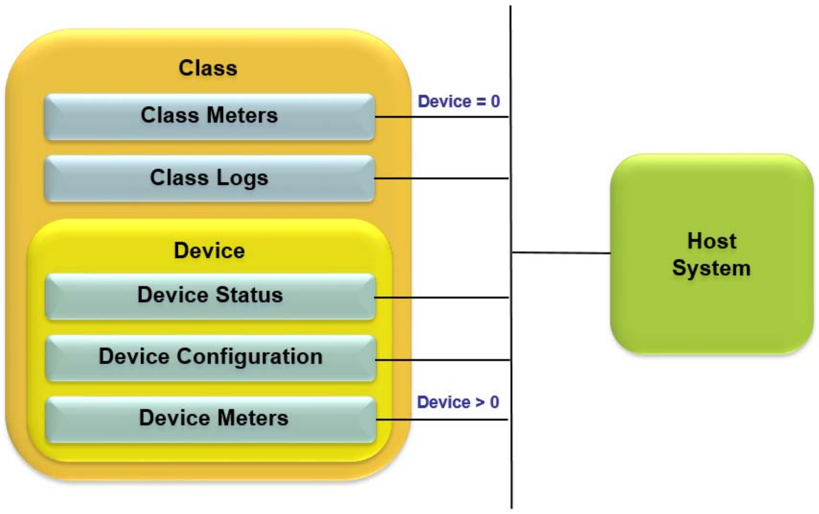
A.1.1 Classes

A *class* identifies a basic unit of functionality in G2S. An EGM consists of a collection of classes. Some classes represent physical devices - cabinet, hopper, printer, etc. Some classes represent logical devices - game play, handpay, progressive, bonus, etc.

A.1.2 Building Blocks

Classes have common characteristics. They are built around a series of standard components, or building blocks; meters, configuration, status, and logs. These four components are used to build the data model for the class. An EGM must persist all critical protocol-related data, including configurations, meters, status, profiles, and transaction logs associated with a device, so that, following an outage, an EGM may return to the same state that it was in prior to the outage. Refer to the G2S Message Protocol documentation for details.

The image below illustrates the logical structure of these four building blocks. Events are used to report changes to the data model. Commands are used to access or change the data model.



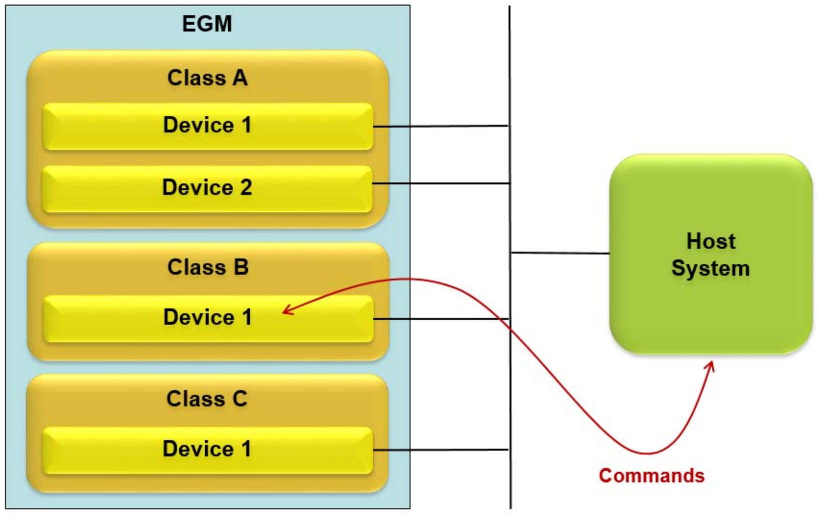
A.1.3 Devices

A *device* is an instance of a class. Commands are directed at devices, not classes. Events are generated by devices, not classes. Status, configuration, and meter information is reported by device. Most classes support multiple devices. The same devices must exist before and after an outage.

A.1.3.1 Commands

Each class contains a set of commands that are used to manage the devices within the class. Some commands are originated by the EGM, such as handpay requests, voucher redemption requests, and progressive hits. Some commands are originated by the host system, such as commands to award bonuses, change configurations, and display messages to players.

The image below illustrates the relationship between different classes and devices in each EGM and the host system. Each class contains standard commands that can be used to query status, configuration, and log information. Each class also contains specialized commands that are used to manage the unique functionality of the class.



Two special classes are used to report meters and events for all classes. The `meters` class is used to report meter information. The `eventHandler` class is used to report events.

A.1.3.2 Status

Most classes include standard *status* information; enabled, locked, configuration identifier, etc. Some classes include additional status information, such as tilts, faults, player card number, etc. Statuses are always reported at the device level.

A.1.3.3 Configuration (profile)

Protocol-related configuration options are contained in the device *profile*. Configuration options can be set locally at the EGM, remotely through a configuration server, or permanently by the manufacturer. Configuration options are always reported by device.

A.1.3.4 Meters

Meters are maintained by device and by class. Class meters contain the life-to-date totals for all devices in the class. Device meters contain life-to-date totals for the device. Three types of meters are used: monetary, counts, and percentages. The maximum monetary meter value is $9,999,999,999.99.

A.1.3.5 Credits

The EGM uses three types of credits.

- *Cashable* - credits that are considered the player's money; can be cashed out of the EGM.
- *Promotional* - credits that were awarded to the player for promotional purposes; can be cashed out of the EGM.
- *Non-Cashable* - credits that were awarded to the player for promotional purposes; cannot be cashed out of the EGM.

An expiration date can be associated with non-cashable credits. The three types of credits are always metered separately.

Credits are consumed in the following order during game play: non-cashable, promotional, and then cashable. The player's money is used last.

A.1.3.6 Transaction Logs

Transaction logs are used to record critical transactions, such as game outcomes, vouchers, and player sessions. Each class contains its own transaction log. All devices within a class share a common transaction log. The size of the log varies by class. Typically, the 35 most-recent transactions are stored. Each transaction is assigned a unique identifier. The same transaction identifier is reported to all systems.

A.1.3.7 Events

Events report changes to the data model. There are only a few events that have no associated changes in the data model, for example: cash-out button pressed.

Events can include affected data. The status, meter, and log information affected by an event can be sent with the event. You have to poll for configuration changes. Each event is assigned a unique identifier. The same event identifier is reported to all systems.

A.1.3.8 Subscriptions

Subscriptions are used to configure the EGM to automatically send event and meter information to the host system in real-time. Hosts can subscribe to particular events and, when the events occur, the EGM will send the events to the host. Hosts may also subscribe to EOD and/or periodic meter information. Subscriptions provide a means for a host to filter out unwanted data from an EGM and only be notified of information required by the host's application.

A.1.3.9 Four-Step Transaction Model

Communications between the EGM and a host typically is in the form of a standard four-step transaction model. Central to this transaction model is the concept of a command request followed by a command response. A typical command sequence will contain four basic steps: request, authorize, commit, and acknowledgment.

A.2 G2S Message Handling

Message Handling in G2S occurs at two levels. These are:

- The Message Level
- The Application Level

A.2.1 Message Level

The *Message Level* is responsible for sending and receiving G2S messages. A G2S message may contain one or more application-level commands. The recipient must either acknowledge delivery of the G2S message or report a message-level error to the sender. The Message Level is only concerned with command delivery. It does not guarantee when, or even if, the commands will get processed.

A.2.2 Application Level

The *Application Level* is responsible for the processing of commands contained within G2S messages. Guaranteed processing is handled at the application level, not the message level. Application-level responses are used to acknowledge the processing of commands. Application-level retries may be necessary to assure that commands are processed. Commands may be processed in any order provided that application integrity is not jeopardized. Commands can be dispatched and prioritized.

A.2.2.1 Asynchronous Processing

G2S uses an asynchronous processing model. That is to say, there is no requirement that the contents of one message be processed before the next message is sent. Hosts and EGMs do not block while waiting for application-level responses.

The host and the EGM must maintain two command queues:

- A queue of inbound commands.
- A queue of outbound commands.

The queues do not need to be persisted. Recovery is done at the application-level.

This diagram below illustrates Message Handling with G2S.

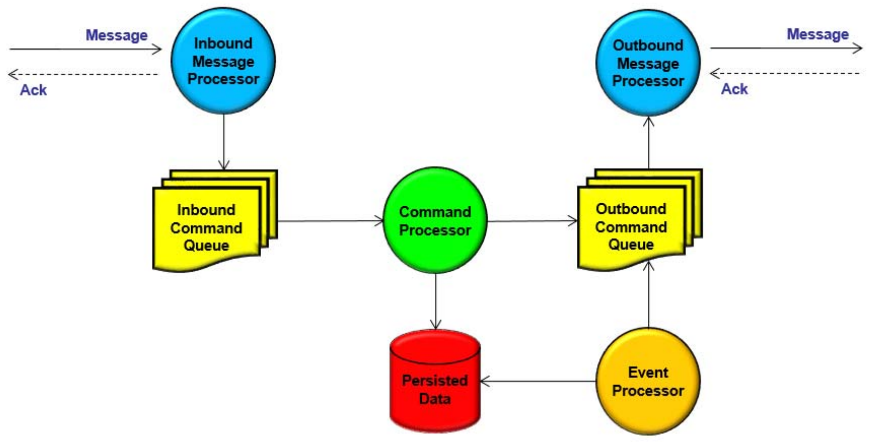
In the upper left corner, messages are being received by an Inbound Message Processor. The messages may contain one or more commands. The Inbound

Message Processor is responsible for unpacking the messages and putting the commands in an Inbound Command Queue for processing.

The SOAP Transport is a request-response protocol, so an acknowledgement must be sent back to the originator of the message. The acknowledgement indicates that the message was received and the next message can be sent. It does not mean that the commands in the message were actually processed, only that they were received.

With WebSocket Transport, the acknowledgement is optional. The WebSocket protocol is not a request-response protocol, so message acknowledgements do not need to be sent. With WebSockets, you can just keep sending messages.

In the center of the diagram is the Command Processor. The Command Processor is responsible for taking commands out of the Inbound Command Queue and generating responses.



The *command processor* is where the application-level processing takes place. The command processor can prioritize and dispatch commands if necessary. The responses are placed in an outbound command queue.

While processing commands, the command processor may also need to access the data model in persistent memory — extracting information or making updates.

The *outbound message processor* is responsible for taking commands out of the outbound command queue and delivering them to the other end-point.

In the lower right corner is the *event processor*. While commands are being received and processed, other events may be happening. These events may cause updates to the data model in persistent memory as well as commands for the other end-point. Those commands will be placed in the *outbound command queue*, intermixed with the commands generated by the command processor.

A.3 G2S Connection Model

A.3.1 Multiple Hosts

An EGM can be configured to communicate with multiple host systems. Each host system must be registered with the EGM. The EGM initiates communications, not the host system. Host systems can be registered locally at the EGM or remotely by a configuration server. An EGM must not communicate with unregistered hosts using the G2S protocol. An EGM can communicate with other hosts using other protocols.

A.3.2 Owners

Every device must have exactly one owner. The owner has command and control authority over the device. The owner is the only entity that can cause the EGM to take an action.

The EGM must only accept "control" commands from the owner of the device.

A.3.3 Configurators

Every device must have exactly one configurator. The configurator can also be the owner. If a host system is the configurator, the host can configure the device remotely. The device can also be configured locally at the EGM.

The EGM must only accept option configuration commands from the configurator of the device.

A.3.4 Guests

A host that is not the owner of a device can be a guest of the device. A device can have multiple guests. A guest can monitor a device, but cannot "control" the device. The EGM must only accept "non-control" commands from guests of the device. The table below outlines functionality by type of host.

Functionality by Type of Host

Host	Control Commands	Configuration Commands	Non-Control Commands
Owner	YES	no	YES
Configurator	no	YES	no
Guest	no	no	YES

Appendix B

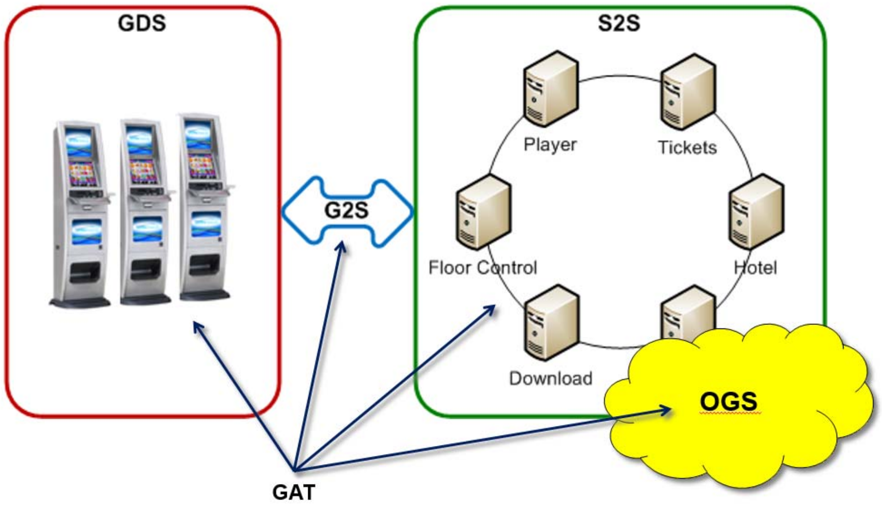
This Appendix contains high-level descriptions of IGSA’s standards.

IGSA’s suite of protocols can be broken down into six categories.

- G2S – Game-to-System
- GAT – Game Authentication
- GDS – Gaming Devices
- S2S – System-to-System
- OGS – Online Gaming
- XPT – Transport

B.1 IGSA Suite of Protocols

The illustration below demonstrates how the various standards interact. Communications between EGMs and peripheral devices is handled by GDS. Communications between EGMs and back-end systems is handled by G2S. Communications amongst back-end systems is handled by S2S. And, communications amongst online gaming systems is handled by OGS. GAT can be applied to all types of endpoints: peripheral devices, EGMs, and systems. XPT (not shown) specifies how G2S and S2S messages are delivered and secured.



B.1.1 G2S — Game-to-System

G2S is a communications protocol that unlocks the power of networked gaming and revolutionizes the way information is exchanged between Electronic Gaming Machines (EGMs) and back-of-house systems (hosts). The protocol enables many advanced features such as software download, remote configuration, remote software verification, and a native embedded player user interface (PUI), which are completely new features for most protocols, as well as for many EGMs.

G2S is significantly different than other protocols, such as Slot Accounting System (SAS), which were not designed to communicate directly to a host system. With other protocols, an interface board is typically inserted between the EGM and the host system. With G2S, everything is done by the EGM. There is no need for interface boards. EGMs communicate directly to host systems.

Two companion specifications help standardize the software download and installation processes.

- *Package Manifest File Format* (PKG): This specification standardizes the format in which software packages are delivered to download servers. It allows a single download server to manage the download and installation process for software from multiple manufacturers.
- *Peripheral Manifest File Format* (PRF): This specification standardizes the format in which peripheral firmware is delivered to EGM manufacturers for integration into downloadable software packages. It provides peripheral device manufacturers with a single method for distributing firmware to EGM manufacturers.

The requirements for native PUI implementations are broken out into two additional companion specifications.

- *Player User Interface* (PUI): This specification describes the requirements for PUI windows. It includes the minimum Flash and HTML requirements for PUI implementations and discusses the isolation of the PUI environment from the EGM. Most importantly, it includes a series of common PUI templates with requirements for deploying those templates on different styles of EGMs.
- *EGM Media Display Interface* (EMDI): PUI content uses this protocol to communicate directly with the EGM through an internal interface, allowing meters, events, and status information to be passed between the EGM and PUI content.

B.1.2 GAT — Game Authentication

GAT is IGSA's serial communication protocol for identifying and authenticating gaming software and firmware. Used by regulators and operators, GAT allows a master to connect to an EGM via a serial cable and authenticate the software and firmware components within the EGM. Typically, a laptop PC is used as the master.

Three additional specifications help extend the power of GAT and broaden its appeal.

- *Network GAT Interface Specification* (NGI): This specification makes GAT functionality available through a standard HTTP network connection. The GAT functionality is the same; the method for accessing it is different.
- *Trusted GAT Results File Format* (TGR): This specification describes a simple file format for distributing expected GAT results in a secure manner. Download servers can use the expected GAT results to verify that software was installed correctly.

- *Certification Database Interface* (CDI): This specification addresses the data interchange needs of regulators, test labs, and suppliers. Based on HTTP/REST and JSON, the specification describes a standard interface for exchanging product approval information amongst regulators, test labs, and suppliers. Future releases will address additional needs, such as product shipments.

B.1.3 GDS — Gaming Device Standards

GDS is IGSA's suite of USB-based protocols for connecting EGMs with peripheral devices such as printers, note acceptors, coin acceptors, hoppers, touch screens, and card readers. GDS also includes the Printer Definition Language (PDL), which is used to define and access templates for printed documents.

The suite of GDS protocols include:

- GDS Printer
- GDS Page Description Language
- GDS Bar Coded Ticket
- GDS Note Acceptor
- GDS Card Reader
- GDS Hopper
- GDS Coin Acceptor
- GDS Touch Screen
- GDS Media Window
- GDS Connector

B.1.4 S2S — System-to-System

S2S helps untangle the jumbled web of back-of-house network interfaces. S2S allows manufacturers to develop a single peer-to-peer communications interface for gaming and non-gaming systems, simplifying connectivity with business partners over wide-area and local-area networks. S2S standardizes information communication and improves consistency to increase operational efficiency.

S2S extends functionality available through G2S – such as progressives, bonuses, and vouchers - to other types of end-points, such as table games and self-service kiosks. S2S allows transactions generated by multiple front-end systems (or from multiple venues) to be dispatched to central application servers in a standardized manner, allowing functionality such as vouchers, wagering accounts, and daily accounting to be consolidated in a single location.

S2S is fully aligned with G2S, making it easy to dispatch G2S transactions and to extend G2S functionality to other end-points.

- *Simple System Interface* (SSI): This specification is designed to address the needs of implementers who do not need the power of S2S's peer-to-peer architecture and prefer a simpler client-server mode of operation. SSI uses an HTTP/REST communications interface and JSON message encoding. It provides implementers with a simple and efficient method for accessing system resources.

B.1.5 OGS — Online Gaming

The *Third-Party Game Interface* (TPI) is IGSA's first communications protocol specifically designed for online gaming. This specification, which is based on JSON, HTTP/REST, and WebSocket technology, describes a standardized interface between iGaming Platforms, Remote Game Servers, and Progressive Jackpot Controllers. It includes functionality for launching games, recording monetary transactions, posting progressive contributions, awarding progressive jackpots, reconciling interrupted games, etc. The specification fully supports online gaming operations that service multiple operators and jurisdictions, allowing the activity associated with each operator or jurisdiction to be easily isolated and reported.

B.1.6 Transport — Point-to-Point and Multicast

A common set of network-based protocols are used to transmit G2S and S2S messages between end-points in a safe and secure manner. Both point-to-point and multicast communications are supported. Whenever possible, the protocols rely on widely adopted Internet technologies, such as HTTP, SOAP, WebSockets, and TLS.

The transport protocols include:

- Network & Security
- Transport Negotiation
- Point-To-Point SOAP/HTTP Transport
- Point-To-Point Websocket Transport
- Multicast Transport

www.igsa.org