



RST
RGS
RPA

System Tester, G2S Scope and
Protocol Analyzer
Quick Start

Looking for more information?

Radical Blue Gaming, LLC.

Questions? Issues? Suggestions? We have lots ways for you to get assistance.

phone: +1-312-897-3737
website: <http://www.radblue.com>
e-mail: support@radblue.com

Gaming Standards Association

If you need a copy of the Gaming Standards Association's latest G2S protocol document, or you want to find out more about the GSA and the work being done by many gaming companies in the areas of protocol standardization for the gaming industry, we encourage you to discover more about the organization.

website: <https://www.gamingstandards.com>

Copyright © 2009-2018 Radical Blue Gaming, LLC. All rights reserved.

All trademarks used within this document are the property of their respective owners. No part of this work may be reproduced in whole or in part, in any manner, without the prior written permission of Radical Blue Gaming, LLC.

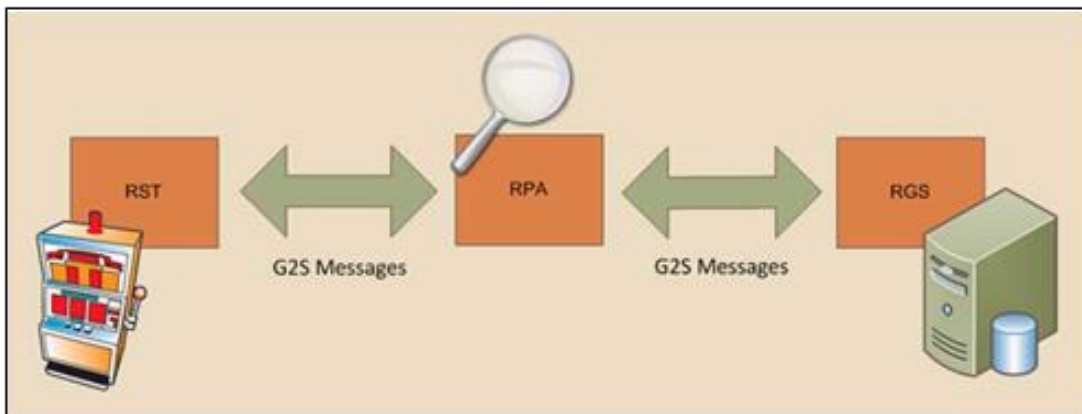
Contents

Introduction to RadBlue Tools	1
Before You Begin.....	2
Additional Resources	2
Module 1 Install the RadBlue Tools.....	3
Module 2 Let's Get Started!.....	4
Activity 2-1: Launch the RST.....	5
For Windows Systems.....	5
For Linux Systems.....	5
RST is Started.....	5
Activity 2-2: Launch the RGS.....	6
Activity 2-3: Getting Around in the Tools.....	7
An Overview of the RST Interface	7
An Overview of the RGS Interface	12
Activity 2-4: Configure the Transcript for Real-Time Updates (RGS and RST)	15
Activity 2-5: Start the SmartEGM (RST)	16
Activity 2-6: View Messages in the Transcript (RST and RGS)	17
What Are You Looking for in the Transcript?.....	17
View the details of a G2S Message in the Transcript.....	18
Flying Solo Activity 2-1	21
Activity 2-7: The commsOnline and descriptorList commands	23
Module 3 Creating Activity	26
Flying Solo Activity 3-1	26
Activity 3-1: Subscribe to Events and Meters (RGS)	27
Flying Solo Activity 3-2	30

Activity 3-2: Send G2S Commands to the EGM (RGS).....	31
Activity 3-3: Create EGM Activity (RST).....	33
Activity 3-4: Exploring G2S Events	36
Flying Solo Activity 3-3 – Player Activity at the EGM (RST).....	39
Flying Solo Activity 3-4 – EGM Device Tilts (RST).....	40
Module 4 Configure the Start-Up Algorithm.....	41
Flying Solo Activity 4-1	41
Activity 4-1: Configure Startup for Disabled Communications	42
Activity 4-2: Start EGM with Disabled Communications (RGS).....	45
Activity 4-3: Enable EGM Communications (RGS).....	46
Flying Solo Activity 4-2	47
Module 5 Adding the Protocol Analyzer (RPA)	48
Flying Solo Activity 5-1	48
Activity 5-1: A First Look at the RPA.....	49
Activity 5-2: Configuring the RPA – a Quick Overview	50
Activity 5-3: How to get RPA in the middle?.....	52
Activity 5-4: View G2S Data in the RPA.....	54
Activity 5-5: Invalid G2S Messages (RGS).....	56
Module 6 Advanced Skills	58
Activity 6-1: Run RST and RGS on Two Different Computers.....	59
Activity 6-2: Add RPA between RST and RGS (again).....	61
Activity 6-3: The Advanced Transcript Analyzer (ATA)	62
Activity 6-4: Exploring the EGM Data Model (RGS)	65
Flying Solo Activity 6-1	69
Activity 6-5: Scripting the Host Engine (RGS).....	71
Flying Solo Activity 5-1: The Balanced Meters Report (RGS)	76
Conclusion - The wrap-up	78

Introduction to RadBlue Tools

The RadBlue G2S Scope (RGS – our G2S Host Simulator) and RadBlue System Tester (RST – our G2S EGM Simulator, also called the SmartEGM) exercise the G2S implementation in an EGM or a host system. You can easily configure the RST and RGS to communicate with one another (on the same computer or on two different computers) to better understand the Gaming Standards Association’s (GSA’s) Game To System (G2S) message protocol. Once the RGS and RST are communicating, you can add the RadBlue Protocol Analyzer (RPA) in the middle of the two applications to give you a window into what’s going on in the message stream between the G2S end-points.



G2S message flow between RST, RPA and RGS.

The purpose of this guide is to get you up and running on the RST’s SmartEGM, RGS and RPA, teaching you the basics of G2S and our tools, and we’ll also touch on some more advanced topics to help you move forward with your G2S implementation. Whether you are developing a new G2S application, testing an existing application or just learning about G2S, this guide will give you the information you need to get started quickly.

We’ll start with the RST and RGS. Once these are up and running, we’ll add the RPA between the two applications and show you how you can use it to troubleshoot messaging issues. Finally, we’ll explore some of the more advanced features of the tools.

Before You Begin

At a minimum, your computer must meet the following requirements to run all three applications simultaneously:

- Operating System: Windows Vista or newer, or Linux
 - Memory: 4 GB
 - Disk Space: 750 MB
 - Processor: Intel 2.8 GHz or comparable
- You should have received a license for each tool from RadBlue. If you haven't already done so, save the attachments to your desktop or a convenient folder where they can be easily located while installing the tools.

No license files? Contact our support team at +1-312-897-3737 or via support@radblue.com.

- Download the product installers from the RadBlue web site: www.radblue.com/products/ (this guide is based on Version 42 of our products)
- Make sure you are familiar with the Gaming Standards Association (GSA) *G2S Message Protocol*. At a minimum, you should read chapters 1 and 2 for an overview of the general rules of the protocol, and a guide to communications between and Electronic Gaming Machine (EGM) and a G2S Host. The *G2S Message Protocol* can be downloaded from the [GSA web site](#), if your company is a member of GSA.

Additional Resources

The RadBlue web site contains information on all RadBlue products, including overviews, release notes, bulletins and user guides. Product user guides contain detailed information, reference material and step-by-step procedures.

- [RGS User Guide](#)
- [RPA User Guide](#)
- [RST User Guide](#)

Module 1

Install the RadBlue Tools

1. Verify the location on your computer of the licenses for RGS, RST and RPA.
2. If you haven't done so already, download the RST, RGS and RPA installers from the RadBlue web site: www.radblue.com/products/. This guide is based on Version 42 of the tools, so it's best to be using that version, or newer one, to minimize the differences.
3. Click on each installer, and then just follow the instructions. Just accept the default file locations for now, and make sure you select the proper license file for each tool.

That's it. If you have any questions or issues, please let us know: support@radblue.com

Module 2

Let's Get Started!

In this module, you will learn how to:

- launch the RST (our EGM simulator)
- launch the RGS (our Host simulator)
- navigate the RST and RGS user interfaces
- configure the Transcript for real-time updates
- load a configuration file, and then start the SmartEGM (RST)
- work with the Transcript, learning how to filter the output to find what you need
- examine the details of the `commsOnline` and the `descriptorList` commands to learn more about two of the most important commands in G2S

Activity 2-1: Launch the RST

For Windows Systems

When each RadBlue tool is installed, an icon is created on your desktop, and a group for that program (**System Tester** in this case) is created in the **RadBlue Tools** folder of the **Start Programs** menu.

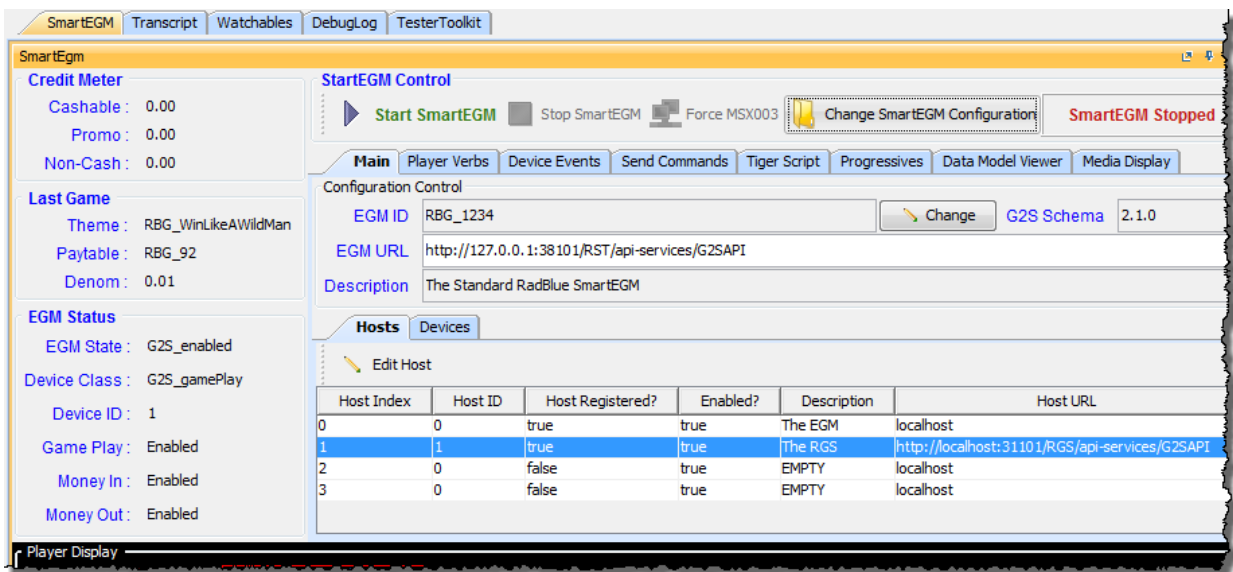
The easiest way to launch the RST is to double-click the RadBlue System Tester icon on the desktop.

For Linux Systems

No **Start Programs** group here, but a RadBlue System Tester icon is created on the desktop. Just click on that icon to launch the RST.

RST is Started...

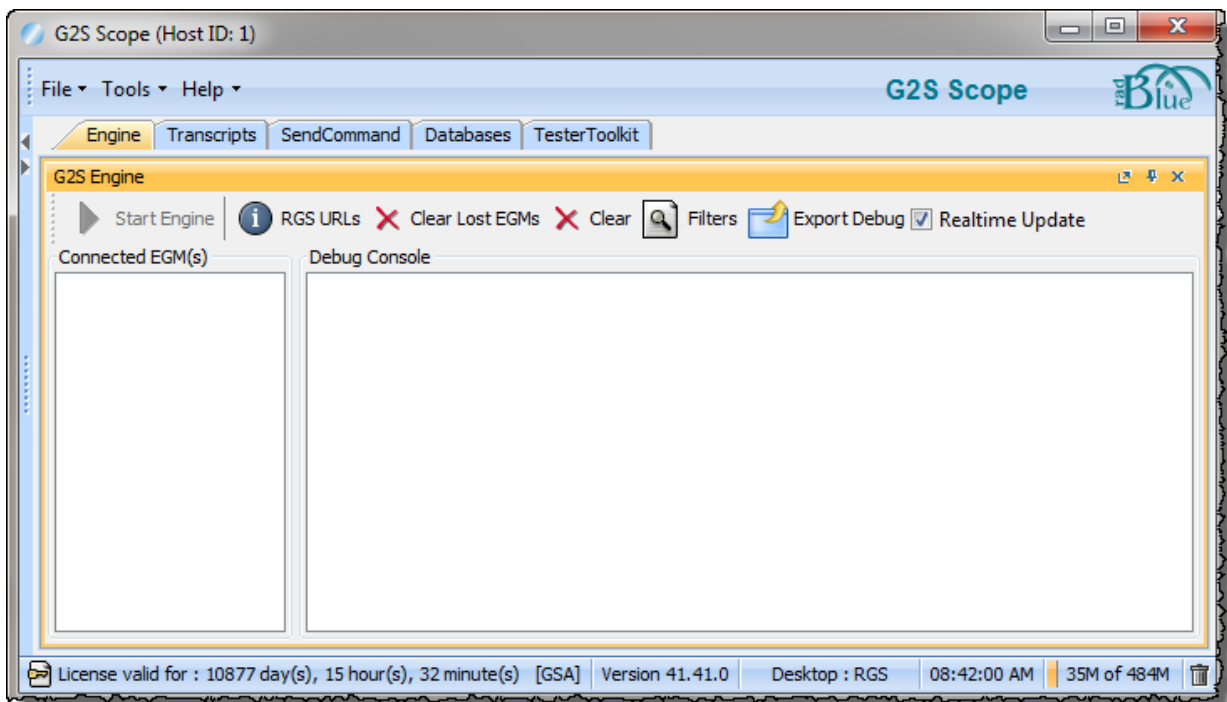
Once you've launched the RST, you should see the **SmartEGM Control** screen (shown below), which you'll soon use to start the G2S conversation between the RST and the RGS:



Activity 2-2: Launch the RGS

Double-click the RadBlue G2S Scope icon to launch the RGS. (A **G2S Scope** directory has also been created in the **RadBlue Tools** folder of the **Start Programs** menu).

Once you've launched the RGS, you should see the **G2S Engine** screen (shown below), which you'll use to observe communications when the RST connects to the RGS:



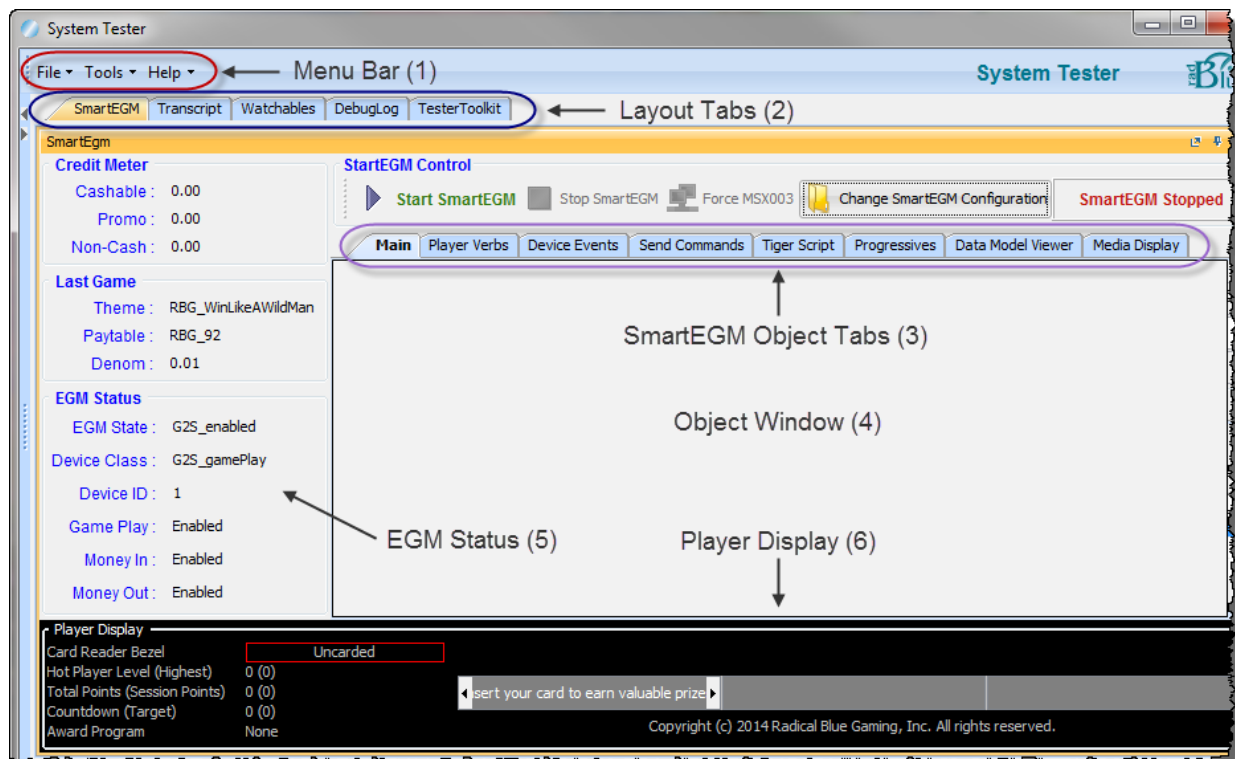
At this point, you should try to fit the two tools side by side (or on two separate screens, if you happen to be lucky) so you can see the activity on both the host and EGM. If you don't have the desktop real estate, you can overlap the applications and toggle back and forth to complete the activities and tasks.

Activity 2-3: Getting Around in the Tools

Let's take a moment or two to review the RST and RGS interfaces.

An Overview of the RST Interface

The SmartEGM layout lets you perform a wide variety of activities at the EGM and then see the results. The SmartEGM layout contains several object tabs, but let's first take a look at the entire desktop.



The SmartEGM layout is comprised of three static areas:
the Object Window, the EGM Status, and the Player Display.

1. **Menu Bar** - From the menu bar you can access three important functional areas:
 - a. **File** – **Import** and **Export** the RST's **Configuration** (so another instance of the RST can use the same settings), and **Export Debug** – which exports a set of debug artifacts in a debug.zip file that you can send to RadBlue (or others) to review what is happening in your tool.
 - b. **Tools** – View / Modify various tool options (**Configure**) and use the **G2S Message Validator** to validate any G2S message against the schema currently being used by the tool.

- c. **Help** – Access the **RST Help** system, **Contact Us** (links to the radblue.com contact page), and view the general information **About the System Tester** (Version number, license info, copyright info, etc.)
- 2. **Layout Tabs** – A series of tabs on the desktop, used to organize objects by function.
 Within each layout, objects can be placed next to each other, or on top of each other (in which case they are accessed by object tabs).
- 3. **SmartEGM Object Tabs** – The SmartEGM tabs allow you to perform various EGM functions, run scripts to automate EGM activity, and view EGM data. Below is a description of each SmartEGM tab.

SmartEGM Tab	Description
Main	From the Main tab, you can load a SmartEGM configuration file, edit EGM and Host IDs and URLs, view a list of the EGM’s devices, and stop/start the SmartEGM engine.
Player Verbs	<p>Player Verbs allow you to simulate EGM actions (player and employee) with the push of a button. The most common actions are at the top of the object window, and sliding windows are used in the lower portion of the window to access different groups of actions.</p> <p>NOTE: The buttons are greyed out (not available) if the EGM is not connected to a host, or if the affected device is disabled or not present in the EGM’s “data model” (which is provided by the configuration file).</p>
Wagering Account Transfers (A sub-tab of Player Verbs)	<p>The Wagering Account Transfer (WAT) icons are accessed via a vertical slider on the left side of the Player Verbs tab.</p> <p>On this tab, WAT Device 1 represents a host-controlled WAT device (where the host initiates the WAT transfer), and Device 2 is an EGM controlled device (where the player initiates the WAT transfer using the EGM’s GUI). You are led through the steps of doing a WAT transfer with either type of device by the icons that are available to you. The Transfer Funds interface allows you to select the direction of transfer.</p> <p>If you are using the RST SmartEGM with the RGS host tool, the valid player IDs are 12345678, 11111111, and 22222222, and the valid WAT accounts can be easily found (and managed) on the WAT Accounts tab for the player on the Player Database tab on the Databases layout in the RGS.</p>

SmartEGM Tab	Description
Device Events	<p>The Device Events tab lets you create the events associated with physical device tilts. A device tilt may also tilt the SmartEGM if the device is “required for play” (in the SmartEGM configuration file).</p> <p>The individual device event tabs are all quite similar. At the top you can select which <i>deviceId</i> to affect (if multiple devices of this type are configured in the SmartEGM). Below this are a number of actions, some of which cause the device to be automatically disabled by the SmartEGM. Selecting the “Clear All Faults” button clears all selected faults, re-enables the device, and if the device was required for play, will also then cause the SmartEGM to be enabled.</p> <p>As each device is selected, it opens up a new sub-tab in the object window so you can easily move between devices of interest.</p>
Send Commands	<p>The Send Commands tab allows you to send several canned messages from the EGM. These are typically large messages (optionList) or general requests for information that don't fit elsewhere.</p>
Tiger Script	<p>Tiger Scripts allow more testing flexibility by automating sequences of actions in an extended script. You can use the sample Tiger scripts provided or create your own, and even automatically receive e-mail notification of the outcome of scripted tests.</p> <p>The use of scripts of commands is beyond the scope of this guide, but we will be happy to work with you if you're interested in extended testing of a G2S Host system.</p>
Progressives	<p>The Progressives tab displays current progressive information. If the host is sending the <code>setProgressiveValue</code> command, the values on this screen are automatically updated.</p>
Data Model Viewer	<p>The Data Model Viewer (DMV) displays the EGM's data model, so you can see the current settings, status, and meter values for any of the G2S devices. Within this tab, you can also create a snapshot of the current values, and also compare any two snapshots. The comparison feature gives you a way to easily detect any changes that may have occurred in the data model. (There's more on this in Chapter 6).</p>
Media Display	<p>The Media Display window provides a Content Table that displays real-time information about each piece of content that is currently loaded on the EGM. A Display window can be used to display our simple flash based content on the EGM (loaded by the RGS).</p>

- Object Window** – The object window displays content associated with the selected SmartEGM tab

5. **EGM Status** – The EGM Status bar is always on the left side of the SmartEGM layout. This screen displays the current conditions of the EGM. As you use player verbs, device events and WAT transfers to play the EGM, send messages and affect the SmartEGM, the EGM Status information updates in real-time.
6. **Player Display** – The Player Display displays the information that would generally be displayed to the player as a result of messages from the player tracking system. In the RST, it shows messages sent from the host to the EGM, including welcome messages, award messages, session messages, and card-out (“goodbye”) messages, as well as countdown and bonus points which are calculated based on parameters from the host.

When creating messages for the player, the host has the option to use substitution *tokens*, which are special characters that display the current value of pre-defined information (for example, player name or EGM ID) at the EGM. Substitution tokens can be used in any message that is directed to the player, and are described in Appendix E of the *G2S Message Protocol* document.

To assist testing efforts, player information is displayed on the left side of the Player Display.

- **Card Reader Bezel** displays the ID number of the inserted player card. A green border indicates that a player card is inserted, a red border indicates that there is no player card inserted at the EGM (or the card has been abandoned), and a yellow border indicates that the player has their card in, but is not playing.
- **Hot Player Level** displays the current hot player level.
 - **(Highest)** is the highest hot player level the player has obtained. This field resets at card-in and card-out, and does not require a player card-in for hot player determination to occur.
- **Total Points** displays the total player bonus points (initial point balance + current session points). This field is initially populated by the `playerSessionStartAck` command.

Note that if you send a `setPointBalance` command after a player session has started, the initial point balance is adjusted to the new value; the number sent is not added to the bonus point total.

- **(Session Points)** indicates the bonus points accrued for the current session. The current session includes: base point awards, player point awards, generic override points awarded, and points awarded by the host (The individual values are in the player log record). This field is only applicable to carded players.

- **Countdown** displays a player's count until earning a specified number of bonus points.
 - **(Target)** is the number that the player must reach to receive bonus point(s). Since G2S provides the option to count down or up, what you can expect to see in this field will change. This field is populated by the `player.setCountdownOverride` command.

Example:

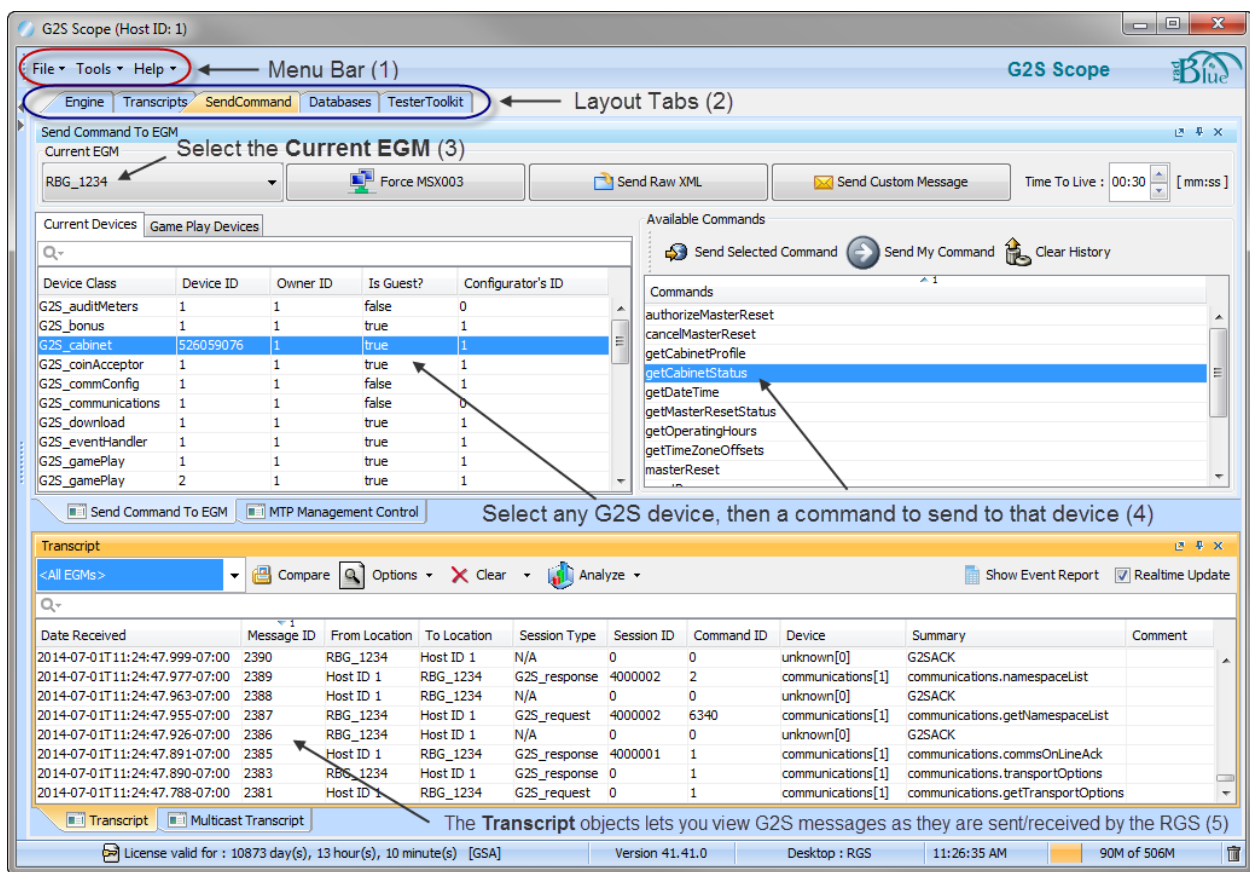
- If you are counting up to 20 with one point earned, you would see:
Countdown (Target) 1 (20)
- If you are counting down from 20 with one point earned, you would see:
Countdown (Target) 19 (0)
- **Award Program** displays the countdown Program currently in effect (base, player or override). Award programs are used to multiple the countdown calculation to celebrate the player's birthday (**player award**), or to encourage play during slow periods in the casino (**override award**).

An Overview of the RGS Interface

From the RGS user interface, you can:

- select one of up to five EGMs and send G2S commands as a host
- view all G2S messages that are sent and received by RGS
- compare any two transcript messages
- view, snapshot and compare instances of the G2S data model for any connected EGM
- work with any of the several databases that are maintained in the RGS (EGM Data Model, vouchers, progressives, players, etc.)
- view SOAP and Multicast message transcripts

There are a lot of things you can do with the RGS, but let's start with an overview of the desktop and the **SendCommand** layout:



1. **Menu Bar** - From the menu bar you can access three important functional areas:
 - a. **File** – Controls are provided by which you can change to an alternate desktop view or modify the desktop on which you are working, by adding new layouts, adding new objects to layouts, etc. - all of which is a bit beyond the scope for a QuickStart guide. You can also **Import** and **Export** the RGS's **Configuration** (so another instance of the RGS can use the same settings), and **Export Debug** – which creates a **debug.zip** file that you can send to RadBlue (or others) to review what is happening in your tool.
 - b. **Tools** – View / Modify various tool options (**Configure**), **Toggle the Floor Tabs** (which shows/hides a directory of objects that can be used on any layout in the tool), and finally you can use the **G2S Message Validator** to validate any G2S message against the schema being used by the tool.
 - c. **Help** – Access the **RGS Help** system, **Contact Us** (which opens the radblue.com contact page), and view the general information **About the G2S Scope** (Version number, license info, copyright info, etc.)
2. **Layout Tabs** - A series of tabs on the desktop, used to organize objects by function. Within each layout, objects can be placed next to each other, or on top of each other (in which case they are accessed by object tabs).

In the standard RGS desktop, there are five Layout Tabs:

- a. **Engine** – Provides a listing of the connected EGMs, access to the Debug Console (where all of the diagnostic messages are printed), along with a button to Export the Debug files so you can share them with others. On this screen, you can also access a listing of the RGS URLs (User Reference Locations – the address that an EGM uses to connect to the RGS on a network).
- b. **Transcripts** – Full screen instances of the three Transcript objects available in the tool. A great place to watch what is going on in the G2S communications between the RGS and your G2S client (RST or an EGM).
- c. **SendCommand** – The objects on this layout (shown above) are used when sending G2S messages to an EGM. A transcript is provided so you can see what's going on without having to switch between Layouts.
- d. **Databases** – A collection of objects that relate to the data used in G2S. The first place to look is at the Data Model Viewer, which shows you all of the current G2S data for any connected EGM. More on this in Chapter 6.

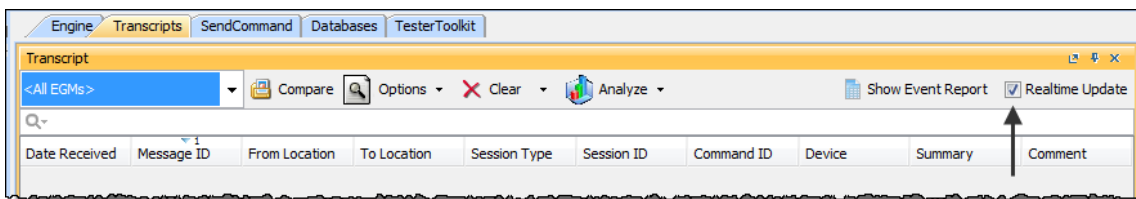
- e. **TesterToolkit** – More advanced functionality that lets you
 - i. tune the **Startup Algorithm** (the series of commands sent by the RGS when a `commsOnline` request is received from an EGM),
 - ii. create **Custom Scripts** – sequences of commands that are sent at the push of a button,
 - iii. work with **Components** – building blocks used in Startup algorithms and custom scripts, and
 - iv. configure the **Response Manager** – control how the RGS responds to G2S requests from an EGM
3. **Current EGM** - Click the drop-down arrow and select the EGM with which you want to work. The RGS supports connections from up to 5 EGMs at one time.
4. **Send Command** - From the Send Command you can send any G2S host command to any of the G2S devices supported by the selected EGM. The **Current Devices** list shows a list of the supported devices, along with the permissions assigned to the RGS. A second tab (Game Play Devices) focuses on the `gamePlay` devices, providing additional information about the game that may be easier to recognize than the `deviceId`. If you double-click on any device, the RGS will provide a pop-up window with additional information about the device.
5. **Transcript** - The Transcript object appears at the bottom of the screen, so you can see the flow of messages between RGS and the RST SmartEGM as you send G2S commands. Since the Transcript is the most important object in the RadBlue tools, we'll devote the next few sections to exercises designed to get you familiar with its functionality.

Activity 2-4: Configure the Transcript for Real-Time Updates (RGS and RST)

The Transcript is a powerful tool for understanding how G2S works. We'll talk more about it in Activity 2-6, but for now, you just need to make sure it's configured correctly.

Do the following on both RST *and* RGS:

1. Click the Transcripts tab, and then make sure the standard Transcript is selected (the first object tab at bottom of the object window).



2. Verify that the **Realtime Update** option is selected.

More Info

Once you select this option, messages appear in the Transcript *as they are received* by the tool.

Changes to the **Realtime Update** setting are persisted, so if you disable this option, it will be disabled the next time you start the application.

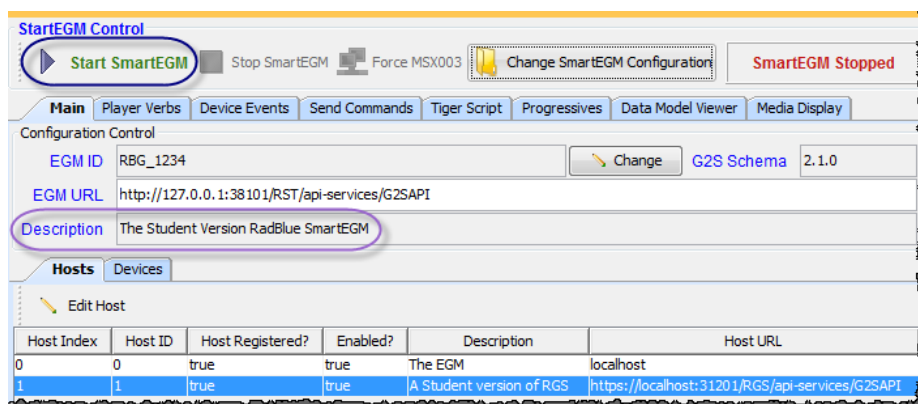
Activity 2-5: Start the SmartEGM (RST)

The SmartEGM configuration file provides the data model for the SmartEGM. By default, a configuration file is loaded for you (typically the last configuration used which persists all meters and settings from the previous session). The “registered host list” (a list of hosts to which the EGM is to connect) is also defined in the configuration file. Depending on the endpoint to which you are communicating, there are several configuration files available to you through the **Change SmartEGM Configuration** option. The SmartEGM file can also be customized to meet your development or testing needs. See the [RST User Guide](#) for more information.

1. Click the **Main tab** on the **SmartEGM** layout, if it is not already selected.

The latest **smartegm-config-gsa.xml** file, which is configured to communicate with the RGS, is automatically loaded for you. If you are using a student license, the **smartegm-config-gsa-student-edition.xml** file should be loaded.

2. Verify that the correct configuration file has been loaded by looking at the **Description** field.
 - If you are using a standard license, the **Description** field displays “The Standard RadBlue SmartEGM”
 - If you are using a student license, the **Description** field displays “The Student Version RadBlue SmartEGM”
3. If necessary, select the correct configuration file.
 - Click **Change SmartEGM Configuration**.
 - Select **smartegm-config-gsa.xml** or **smartegm-config-gsa-student-edition.xml** if you are using a student license, then click **Open** to use this file.



4. Click **Start SmartEGM** to start G2S communications with the RGS.

Activity 2-6: View Messages in the Transcript (RST and RGS)

The Transcript is available in all of RadBlue's G2S Tools and provides a record of the G2S messages that are being sent by each of the G2S endpoints. Thus, your basic workflow in RST and RGS will be:

1. Do something.
2. Look at the Transcript.
3. Do something else.
4. Look at the Transcript.

(You get the idea.)

As soon as you started the SmartEGM, messages began flowing between the RST's SmartEGM and the RGS. In this activity, we'll explore the Transcript.

1. Click the **Transcript** tab in the RST or the RGS.

Date Received	Message ID	From Location	To Location	Session Type	Session ID	Command ID	Device	Summary	Comment
2014-07-02T11:00:56.998-07:00	20189	RBG_1234	Host ID 1	G2S_response	200000	6877	communicat...	communications.descriptorList	
2014-07-02T11:00:56.946-07:00	20188	RBG_1234	Host ID 1	N/A	0	0	unknown[0]	G2SACK	
2014-07-02T11:00:56.932-07:00	20187	Host ID 1	RBG_1234	G2S_request	200000	4	communicat...	communications.getDescriptor	
2014-07-02T11:00:56.729-07:00	20186	RBG_1234	Host ID 1	N/A	0	0	unknown[0]	G2SACK	
2014-07-02T11:00:56.714-07:00	20185	Host ID 1	RBG_1234	G2S_response	18000003	3	communicat...	communications.commsDisabledAck	
2014-07-02T11:00:56.685-07:00	20184	Host ID 1	RBG_1234	N/A	0	0	unknown[0]	G2SACK	
2014-07-02T11:00:56.664-07:00	20183	RBG_1234	Host ID 1	G2S_request	18000003	6876	communicat...	communications.commsDisabled	
2014-07-02T11:00:56.470-07:00	20182	RBG_1234	Host ID 1	N/A	0	0	unknown[0]	G2SACK	

2. Look at the G2S messages flowing in and out of each tool. Notice the direction (**To Location** and **From Location**), G2S command (displayed in the **Summary** column), and the Session fields (**Session Type** and **Session ID**).

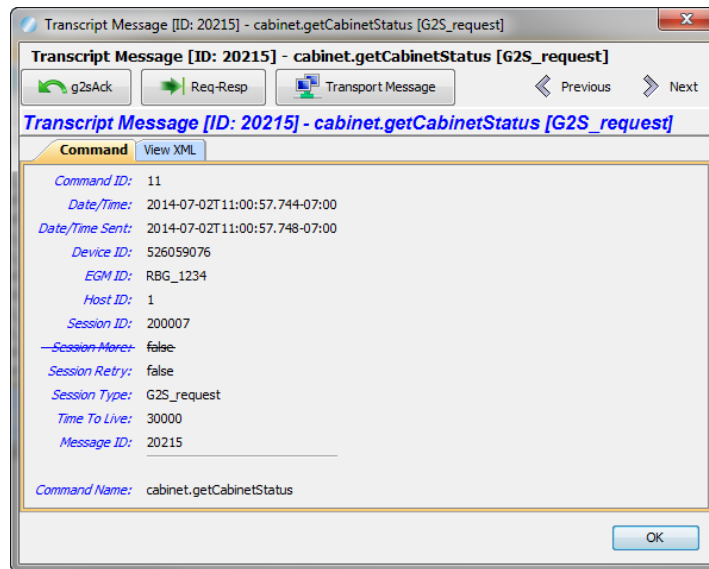
What Are You Looking for in the Transcript?






- Are the correct commands being sent?
For example, is the `commsOnline` command being sent during startup?
- Are messages being acknowledged (with a `G2SACK` as well as a response message (unless Session Type = "G2S_notification"))?
- Are there correct request-response pairs?

For example, if a `communications.getDescriptor` request command is sent, a corresponding `communications.descriptorList` response command should be returned. Note that the **Session ID** is the same for the request and response messages.

View the details of a G2S Message in the Transcript

1. Select any message in the Transcript, and double-click to display the details



2. This view is known as the “command view”, since we have parsed the message for you and are displaying the important fields in an easy-to-read format. In more complex messages, the data is presented in a table that can also be expanded...
3. Starting at the top, you can see that this is a `getCabinetStatus` request, which is a command in the `cabinet` class. The RadBlue tools add a “message Id” to every message to help serialize them in the transcript (shown here as [ID: 20215]).
4. The next row contains several action buttons to help you find information that is related to the selected message:
 - a.  is a link to the `g2sAck` message – the transport acknowledgement sent by the receiver to indicate the message was received and will be processed.
 - b.  is a link to the other message in the request/response pair. As this is a request message, pressing this button will bring up the Counterpart message (the `cabinetStatus` response).
 - c.  is a link to the transport view of the message. Most G2S messages are currently sent using SOAP (**Simple Object Access Protocol**), which is essentially a wrapper for messages sent via web-services. This link pops up a window that shows this message with its SOAP encoding.
 - d.   These buttons are used to move to an adjacent message in the transcript.

5. Scroll down the screen, and look at the content of the message. Note that all details of the message are displayed in an easy-to-read format. **Class** level fields are shown above the horizontal line, and **Command** level fields are shown below the line. In our example above, you can see the various IDs and time stamps used in the message, along with other fields that should be starting to be familiar. Fields that have a strikethrough (like Session More) were valid in an early release of G2S, but have since been deprecated, so they are now ignored by the end-points.

If you now click on the **View XML** tab, the display switches to the XML view of the actual G2S message (the blue and green rectangles were added for this document):

```

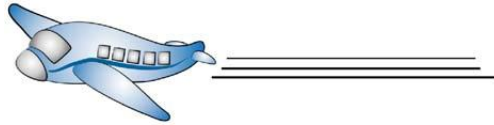
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <g2s:g2sMessage xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3">
3   <g2s:g2sBody g2s:dateTimeSent="2014-07-02T11:00:57.748-07:00" g2s:egmId="RBG_1234"
4     g2s:hostId="1">
5     <g2s:cabinet g2s:commandId="11" g2s:dateTime="2014-07-02T11:00:57.744-07:00"
6       g2s:deviceId="526059076"
7       g2s:errorCode="G2S_none"
8       g2s:errorMessage=""
9       g2s:sessionId="200007"
10      g2s:sessionMore="false"
11      g2s:sessionRetry="false"
12      g2s:sessionType="G2S_request"
13      g2s:timeToLive="30000">
14       <g2s:getCabinetStatus/>
15     </g2s:cabinet>
16   </g2s:g2sBody>
17 </g2s:g2sMessage>

```

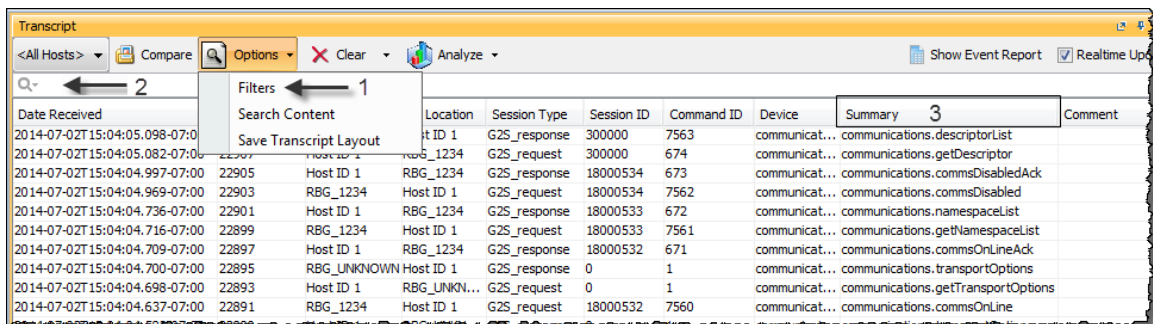
1. First off, since we've added color-coding to the XML, it's much easier to figure out what is what in the message: Purple is for *elements* and wrapper info, black is for *attributes*, and blue is for text strings that are assigned to the attributes of the message.
2. In the top frame, you can choose between the Text View of the XML (what you'll use 99.9% of the time), and a Hex View of the message, which is handy if you're investigating a weird or unprintable character in the message.
3. You can see that a G2S message starts with some standard XML encoding information, followed by a g2sMessage wrapper that provides the base *namespace* of the message, then then a g2sBody wrapper that provides high level routing information (the *egmId* and *hostId*) that can be used to make sure this message has been sent from a known end-point, and it is really meant for this receiver.
4. For G2S applications, the Class information contains the high level information needed to process this message:

- a. *commandId* is a sequential counter that can be used to detect when messages arrive out of order.
 - b. *dateTime* is when the message was actually generated, as compared to *dateTimeSent* in the *g2sBody* element which indicates when this copy of the command was wrapped in a message and sent to the other endpoints web-server.
 - c. The *deviceClass* (*g2s:cabinet*) and *deviceId* indicate which specific device generated this command, or the one for which it is intended. This pairing is important when there are multiple devices in one class in an EGM (a perfect example is *gamePlay* devices).
 - d. The *errorCode* and *errorText* attributes are used when an application error occurs, in which case there must never be a command element in the message.
 - e. The session information is used to indicate whether this is a request or a response (*sessionType*) and the *sessionId* is created by the requestor and returned by the responder to indicate which request is being answered. The other two session attributes are being deprecated, and so can be ignored.
5. Finally, since G2S messages can be quite large, a Find utility is offered in the bottom frame of the control to make it easy to find a specific string in the message.

Flying Solo Activity 2-1



Now that you're familiar with the Transcript, use the Transcript filtering options to search for a specific message or message type.



Task 1 Click **Options** then **Filters**, clear the **Show G2S ACKs** check box, and click **Apply**. How does this change your view of the information?

You can also use this filtering mechanism to focus in on the commands or events for a particular class, and/or for commands relating to one or more specific devices.

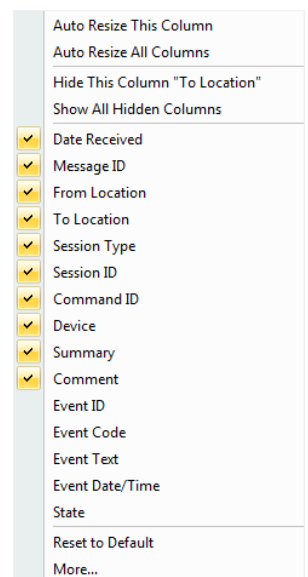
Task 2 Click in the **Quick Search** dialog box (#2 above - the blank bar with a magnifying glass on the left side), and type **comms**. What happens in the Transcript as you type? Click on the magnifying glass, and use the options in the menu to expand and narrow your search.

Task 3 All columns are sortable. Click in the Summary column header to sort the Transcript by command name. (Note: First click is ascending, second click is descending, third click clears the sort).

Task 4 Sort on multiple columns using the CTRL key. Press CTRL and click on the Session Type column header. Then, click Summary. How does this affect how the information sorts?

Task 5 You can choose to show or hide any of the Transcript columns. Right-click in any column header (see picture to the right). Click Summary to hide the Summary column. Right-click again, and click Summary. What happens?

Task 6 Right-click in any column header, and resize one or all columns to fit the content (top entries in the pop-up menu).



- Task 7** Rearrange column order by clicking on a column header and dragging it to a new location.
- Task 8** Once you've rearranged the Transcript to suit your needs, you can save the configuration and it will continue to be used for this tool. Just select **Options > Save Transcript Layout**, and the Transcript Layout will be saved. (Just in case, there's a "Reset to Default" option in the pop-up menu to get you back to the default settings).
- Task 9 Color-coding** - Select the **Show G2S ACKs** check box once again to show the g2sAcks (See Task 1 if you need help). Sort the display in **Date Received** order. Now, select any G2S command in the transcript. You'll see that the tools color code the transcript to show you the Counterpart Message (in blue bold) and the g2sAck (in green bold). This is helpful when lots of commands come in a burst.

Date Received	Message ID	From Location	To Location	Session Type	Session ID	Command ID	Device	Summary
2014-07-02T15:04:04.969-07:00	22903	RBG_1234	Host ID 1	G2S_request	18000534	7562	communicatio...	communications.commsDisabled
2014-07-02T15:04:04.987-07:00	22904	Host ID 1	RBG_1234	N/A	0	0	unknown[0]	G2SACK
2014-07-02T15:04:04.997-07:00	22905	Host ID 1	RBG_1234	G2S_response	18000534	673	communicatio...	communications.commsDisabledAck
2014-07-02T15:04:05.002-07:00	22906	RBG_1234	Host ID 1	N/A	0	0	unknown[0]	G2SACK
2014-07-02T15:04:05.032-07:00	22907	Host ID 1	RBG_1234	G2S_request	300000	574	communications[1]	communications.netDescriptor

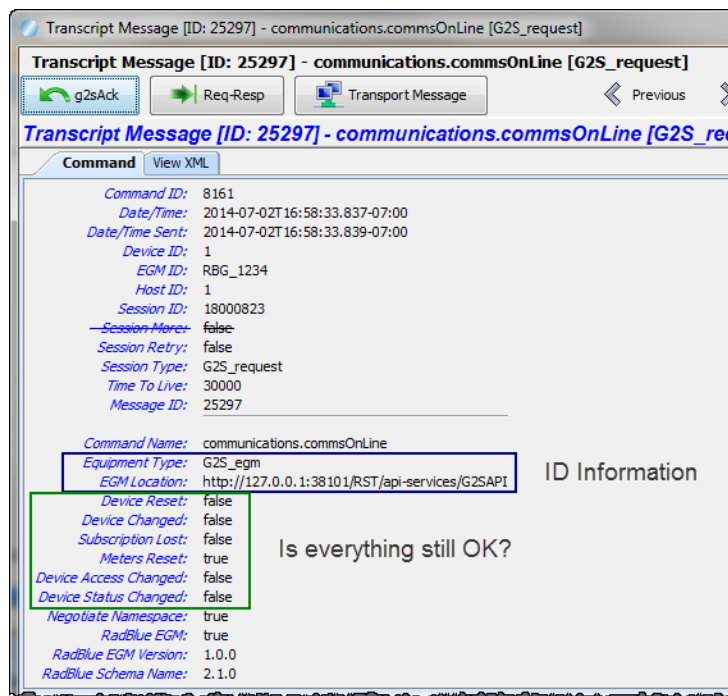
Activity 2-7: The commsOnline and descriptorList commands

In this activity, we'll take a look at the first G2S commands used by an EGM and a system to start G2S communications. To prepare, please do the following:

1. Stop the SmartEGM in the RST (Press the **Stop SmartEGM** button on the SmartEGM layout)
2. On the RGS, switch to the Transcript layout, and then clear the Transcript table by pressing the **Clear** button, so you start with an empty table.

We're now ready to restart communications between the EGM and the Host:

1. On the RST, press the **Start SmartEGM** button to start G2S communications between the SmartEGM and the RGS.
2. On the RGS, look at the Transcript to see the messages in real-time as the host runs its "Startup algorithm" – a sequence of commands that are automatically executed when an EGM first connects to the host. The RGS does this to discover all of the details of the EGM that has just connected.
3. Use the Quick Search feature on the Transcript in the RGS to search for **commsOnline**, which limits the transcript to two messages, the `commsOnline` from the EGM, and the resulting `commsOnlineAck` response from the host.
4. Double-click on the `commsOnline` to launch the detail view of that command:



5. The `commsOnline` is the first G2S command sent from the EGM to the host. Once started, the host patiently waits for EGMs to connect to his web-server (at the address listed as the Host URL in the SmartEGM). The host examines the contents of the `commsOnline` to discover the EGM's ID and the network location (`egmLocation`) by which the host can connect to the EGM's web-server. The `commsOnlineAck` occurs when the host has successfully contacted the EGM's web-server.

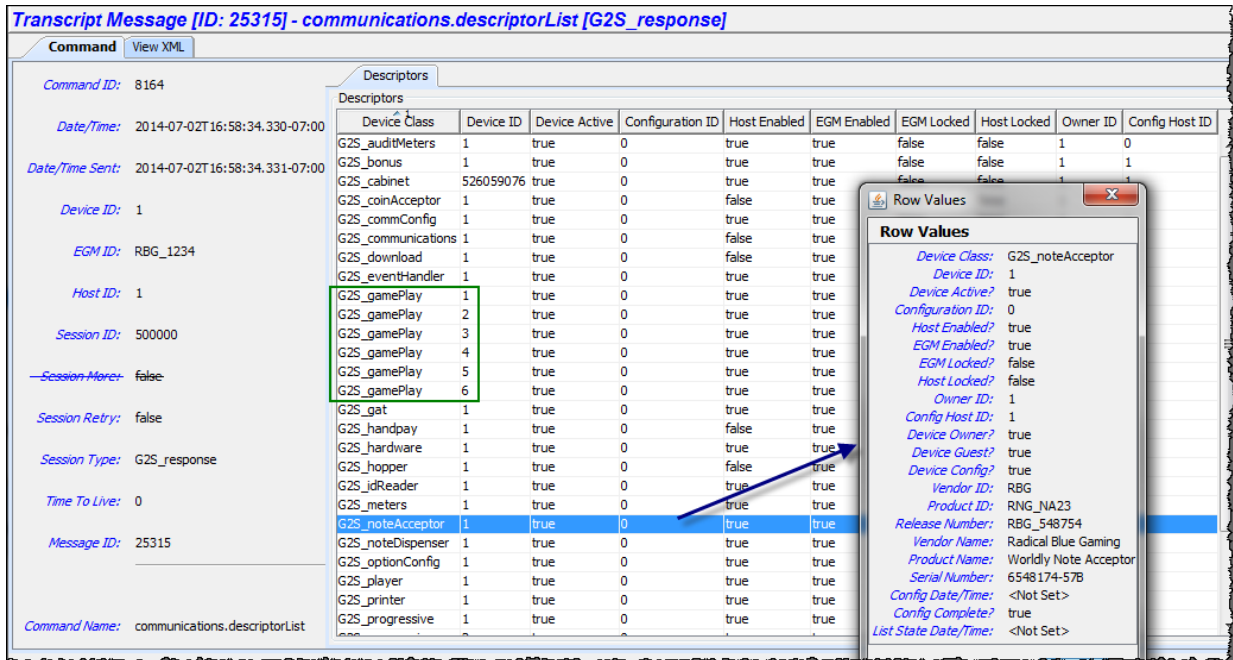
In essence, the EGM has contacted the host to say "I want to do G2S with you", and the host then responds and says "Sounds good to me - we're connected!". Once this connection is established, it continues until the host or EGM shuts down.

However, when an EGM connects to the host, the first thing the host needs to do is to figure out which G2S functionality ("classes") the EGM supports, and what permissions this host has for each of the instances ("devices") of that functionality, so the host knows how he can interact with the EGM. To accomplish this first interrogation, the host requests a `descriptorList` from the EGM, which provides everything the host needs to know.

6. Use the Quick Search feature on the Transcript in the RGS to search for **descriptor**, which limits the transcript to two different messages, the `getDescriptor` request from the Host, and the resulting `descriptorList` response from the EGM.

Note: Though there are a number of attributes in the `getDescriptor` request that can be used to tune which devices are reported, the RGS always requests that all devices in all classes be reported in the `descriptorList`.

7. Double-click on the `descriptorList` to launch the detail view of that command (please see the command view on the next page):



8. Look at the Device Class column to see which device classes are currently supported by RST.
9. Notice how many devices are available for each class. (For example, in the figure above, there are six gamePlay devices – indicated by a green outline.)
10. Review the attributes that are displayed in the table – it's not all of the fields for each device, just those that are most important.
11. Double-click any device to view all of the attributes for that device (in the example above, the noteAcceptor was used). Some devices are *physical* (noteAcceptor), while others are *logical* (communications and download devices).
12. Click Back to close the Row Values screen.
13. Click OK to close the descriptorList detail view and return to the Transcript.

Congratulations, you've completed Module 2! Close both tools and take a well-deserved break.

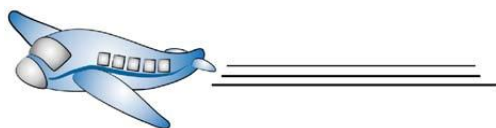
Module 3

Creating Activity

In this module you will learn to:

- subscribe to events and meters
- send G2S commands to an EGM using the Send Command layout in the RGS
- use the SmartEGM's Player Verbs to create EGM activity
- view eventReports and all sorts of related eventHandler information

Flying Solo Activity 3-1



Now that you've completed Module 2, let's test your knowledge to get up and running for Module 3.

Task 1 Launch RGS.

Task 2 Launch RST (but don't start the SmartEGM, just yet)

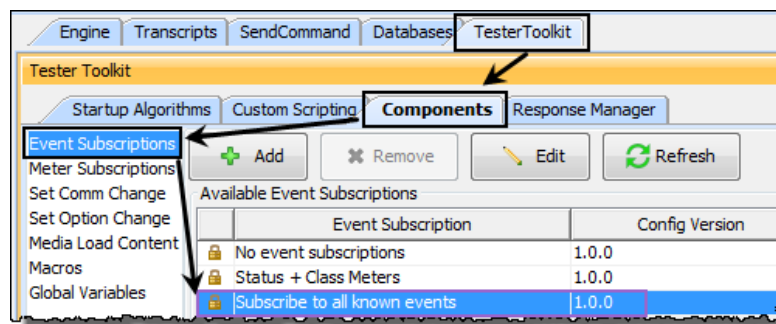
Activity 3-1: Subscribe to Events and Meters (RGS)

Event and meter subscriptions are set by the RGS during the start-up algorithm that is executed whenever a `commsOnline` command is received from an EGM. **Components** are provided in the RGS to create the building blocks that can be used over and over in the start-up algorithm (and in custom scripts).

Event subscriptions allow you to be notified by the EGM whenever something in the data model changes, which causes an event to be generated (if someone is listening). If you have *subscribed* to the event, you will be notified when that event occurs. Meter subscriptions are similar, but are typically time based, causing the EGM to send (or capture in the case of `auditMeters`) the specified set of meters when a time interval (or other activity) that you have identified occurs.

This activity shows you how to view/modify the initial event and meter subscriptions used by the RGS.

1. Select the **TesterToolkit** layout, then the **Components** tab in that layout
2. Select **Event Subscriptions** in the list of component types (on the left) which provides a list of Event Subscriptions that are available to be used in scripts.

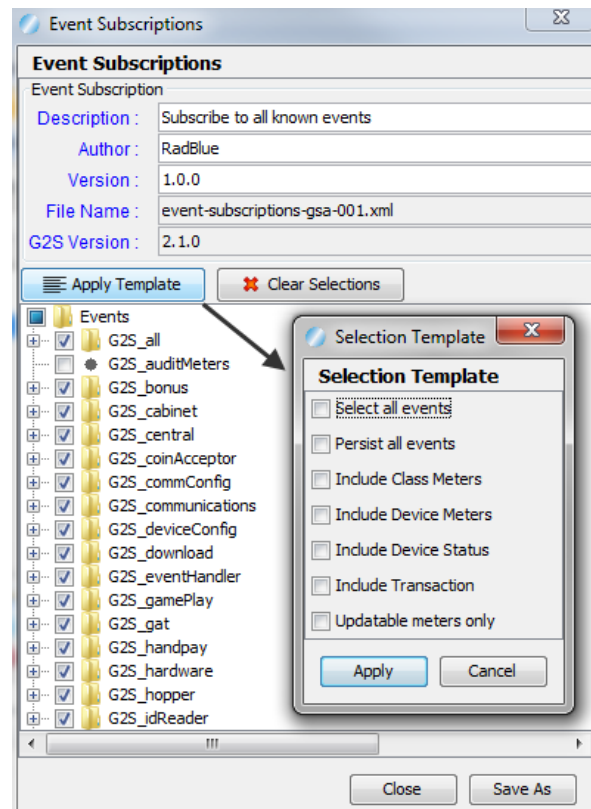


3. Double-click the event subscription called "**Subscribe to all known events**".

The Lock symbol in the first column means that this is a standard component that is distributed with the tool. It cannot be modified, but it can be used as the basis of new components (modified, then "Saved As" to make a new file).

In this tree-structured control, selecting or clearing any box automatically causes any options beneath that box to be affected. The organization of the tree is by Class, then by a listing of all events within the class, followed by the six possible attributes for each event. These attributes control what “associated data” is to be included with the eventReport, whether only updatable meters are sent, and whether the eventReport must be persisted until an eventReportAck is received by the EGM (indicating that the message has been processed by the host).

You can also use a **Template** to make it easier to do global changes of these attributes. First, select the Classes and events that you are interested in, and then use the **Apply Template** button to launch a control that will allow you to select which attributes will be set to true for your set of chosen events.



For simplicity, the selected event subscription is applied during the start-up algorithm to every device within the class, reported by the EGM in its descriptorList. This can be refined later using the Send Command control to set/clear subscriptions to events for a specific device within a class.

4. For this activity, just press **Cancel** to exit the Selection Template control, and then press **Close** to close the event subscription component. We’ll just use the standard event subscription for the examples in this section, but feel free to take a look at the sample event subscription components that ship with the tool.

5. Now select **Meter Subscriptions** in the list of component types (on the left) which provides a list of Meter Subscriptions that are available to be used in scripts.
6. Double-click the meter subscription called “**Standard EOD and Periodic Subscriptions**”.

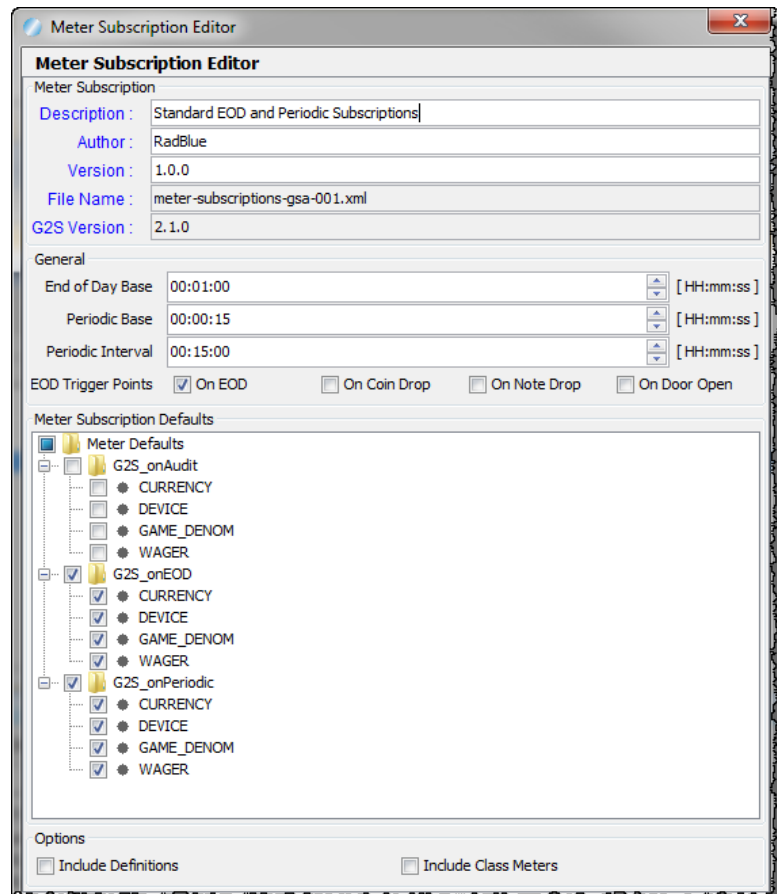
In the General section of the Meter Options screen, you can specify the following:

- **End-of-Day Base** - Indicates when the daily EOD meters are to be sent, or when an audit meter snapshot is to be taken (in time after midnight).
- **Periodic Base** - Indicates when the daily Periodic Meter report schedule begins (also time after midnight).
- **Periodic Interval** - Indicates how frequently Periodic Meter reports are to be generated.

After this you can specify one or more End of Day (EOD) trigger points – On EOD is time-based, and the others are event based.

In the Meter Subscription Defaults section, you can specify the meter subscription(s) that are sent to the EGM during the start-up algorithm for every device in the `descriptorList` command (by definition, the EGM ignores subscriptions for devices for which it does not maintain meters). The available meters sets (currency, device, game denomination, and wager meters) can be selected for the **audit meter**, **end-of-day** and **periodic** meter subscriptions.

In the **Options** section at the bottom of the screen, you can specify whether the meter subscription should also tell the EGM to include definitions of each meter in the reports that are generated by the EGM as a result of these subscriptions. Including the meter

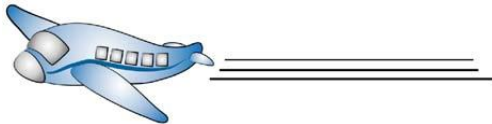


definitions causes three additional attributes to be included with each meter - *meterType* (count or value meter), *meterIncreasing* (Does meter always increase? Or does it go up and down like a credit meter?), and *meterRollover* (the maximum value for this meter after which it resets to zero).

Finally, you can subscribe to the *Class* meters for each G2S class. The Class meters in G2S provide a summary of all activity that has ever take place in all devices in a specific class. This is especially useful in the *gamePlay* class, where individual *gamePlay* devices might be added and removed, so it is handy to have a life-to-date total of all *gamePlay* activity that has occurred on any *gamePlay* device in this EGM.

7. For this activity, just press **Cancel** to exit the Meter Subscription Editor control, and then press **Close** to close the meter subscription component. We'll just use the standard meter subscription for our later examples, but take a look at the sample meter subscription components that ship with the tool to see some of the various combinations that are possible.

Flying Solo Activity 3-2



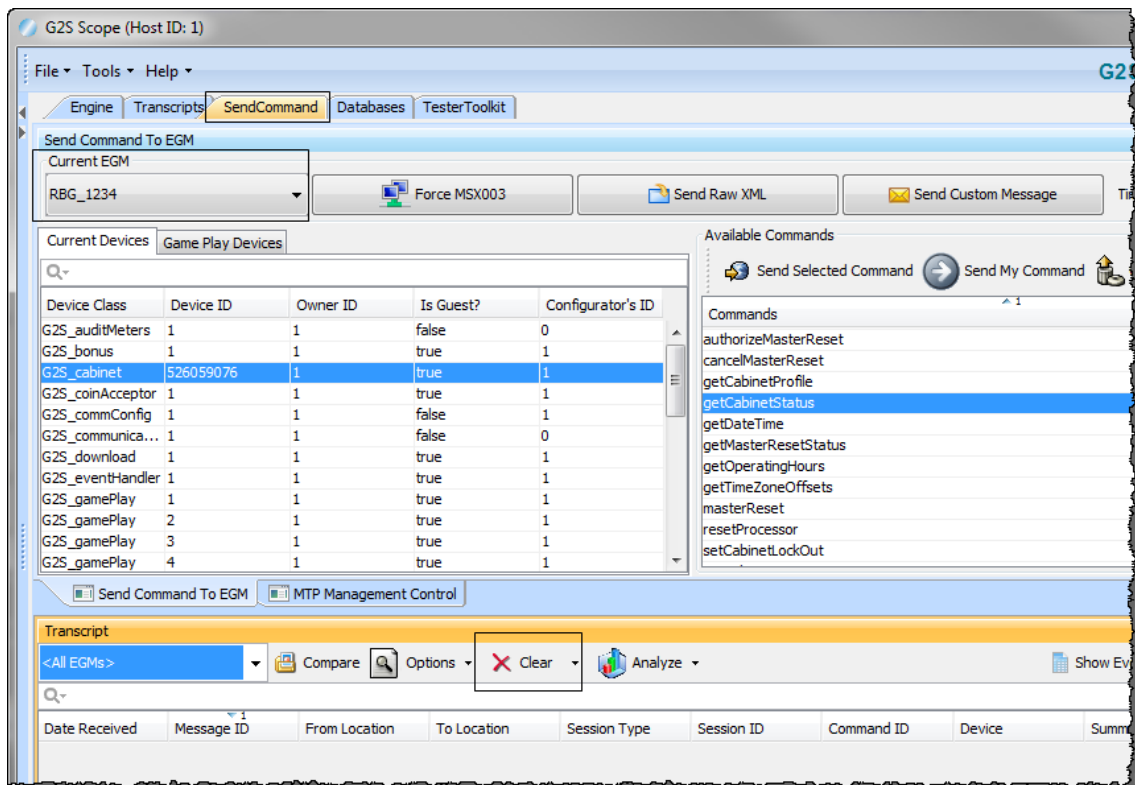
Now that you've reviewed the standard event and meter subscriptions, start the tools.

Task 1 (RST) Start the SmartEGM.

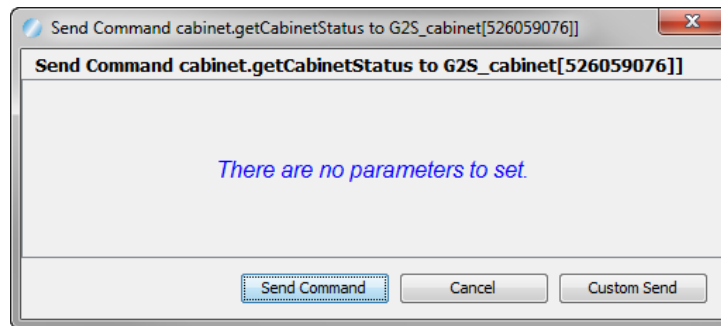
Task 2 (RGS) Look at the Transcript and verify that the SmartEGM is now sending events (an easy way to do this is to do a Quick Search for **event**).

Activity 3-2: Send G2S Commands to the EGM (RGS)

The SendCommand layout in the RGS lets you send G2S commands to an EGM and view the responses. This layout contains the sendCommand object (titled “Send Command To EGM”) and the Transcript.



1. Select the **SendCommand** layout tab.
2. On the **Transcript** object (lower section) click **Clear**, then **Display** to clear this instance of the Transcript object.
3. Select the **Send Command to EGM** object (top section). Notice that the Current EGM drop-down lets you select one of up to five EGMs that may be configured to communicate with RGS at once. Any commands you send will be sent to the EGM identifier displayed under Current EGM. Since you are only communicating with RST, no additional EGM identifiers are displayed.
4. Select **G2S_cabinet** from the Current Devices list. The Available Commands list on the right side is automatically populated with all commands associated with the selected device.
5. Select **getCabinetStatus** from the Available Commands list.



Note If the selected command has any configurable attributes, you will be able to modify those attributes in this pop-up, before sending the command to the EGM.

6. Click **Send Command** to send this command to the EGM.
7. Look at the messages as they appear in the Transcript. Are the `G2SACK` messages displaying? If so, you can use **Options** → **Filters** to remove them from the display by clearing the **Show G2S ACKs** checkbox).

Date Received	Message ID	From Location	To Location	Session Type	Session ID	Command ID	Device	Summary	Comment
2014-07-03T13:01:52.008-07:00	11979	RBG_1234	Host ID 1	G2S_response	200155	8821	cabinet[526059076]	cabinet.cabinetStatus	
2014-07-03T13:01:51.979-07:00	11977	Host ID 1	RBG_1234	G2S_request	200155	463	cabinet[526059076]	cabinet.getCabinetStatus	Via Send Panel #1

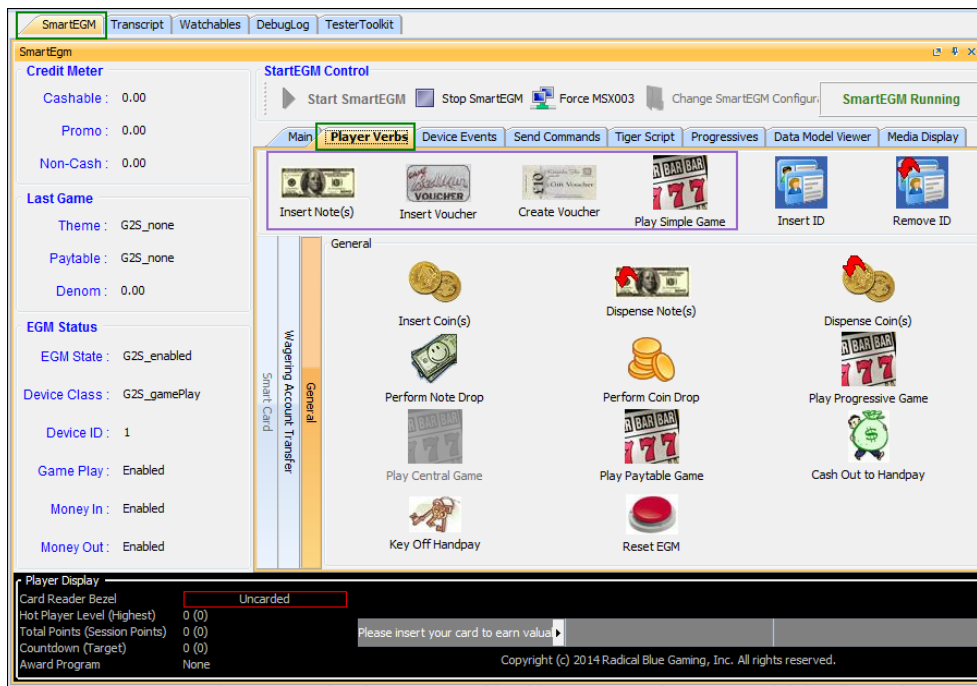
8. Double-click on the `cabinet.cabinetStatus` response, and review the content. This command provides all of the status information about the base EGM (accessed via the **cabinet** class). Look at the G2S spec for information on each of the attributes.

Activity 3-3: Create EGM Activity (RST)

Use the SmartEGM Player Verbs in the RST to simulate player activity at an EGM. Each Player Verb button represents an action at an EGM and may generate multiple messages to RGS. This activity shows you how to simulate simple game play. *Be sure to review the Transcript after you send each Player Verb, so you can see the flow of messages.*

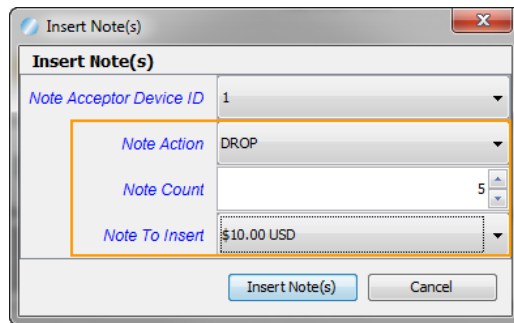
The EGM Status bar appears alongside all SmartEGM screens. The EGM Status displays current EGM settings. As you use player verbs, device events and WAT transfers to create activity that drives G2S messages, the EGM Status information is automatically updated.

1. Select the **Player Verbs** tab on the **SmartEGM** layout of the RST.

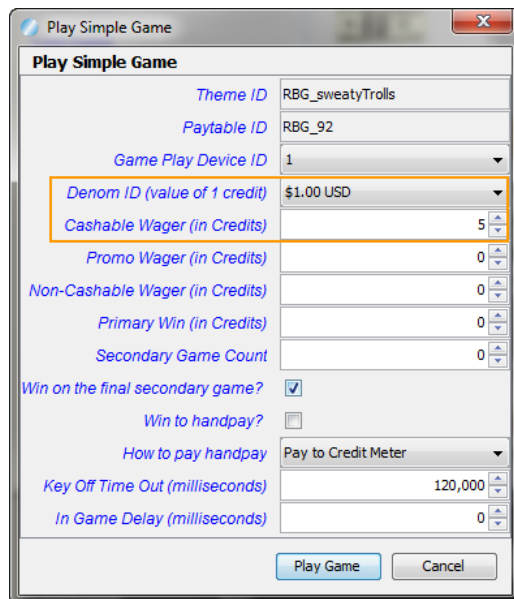


For this activity, you will simulate the following actions at an EGM:

- insert a note
- play a simple game
- create a voucher
- insert a voucher

2. Click **Insert Note(s)**.

3. Change the GUI options so you insert five (5) notes with a denomination of \$10.00 USD, and indicating they should go to the Drop (stacker) rather than being sent to the note dispenser.
4. Click the **Insert Note(s)** button. Notice that the **Credit Meter [Cashable]** field in the EGM Status panel changes from zero (0) to 50.00.
5. Click **Play Simple Game**.



6. Change the GUI options so that you wager \$5 (5 credits that are each worth \$1 USD)
7. Click **Play Game**
8. Look at the **EGM Status** panel (on the left side of the layout). You'll see that the **Credit Meter [Cashable]** field is now **45.00**, the theme, paytable, and denomination of the **Last Game** info have been updated, and the **EGM Status** now indicates that the last gamePlay device that was accessed was G2S_gamePlay device 1.

9. Click **Create Voucher**.

Since the EGM funds in this example are cashable, leave the voucher **Credit Type** as **Cashable**. If the EGM credit meter also has promotional and/or non-cashable, you could select one of those credit types for your voucher. When you select a credit type, the entire amount shown on the EGM credit meter transfers to the voucher, and that meter changes to zero (0.00) in the **EGM Status** panel.

10. Click **Create Voucher**, and you'll see a pop-up that provides some of the details that would be printed on a physical voucher

11. Review the content of the (simulated) voucher, and click **OK**.

The validation ID is stored in the RST's voucher database and becomes available for the **Insert Voucher** verb.

12. Click **Insert Voucher**

The Validation ID field is automatically populated with an ID from the RST's voucher database (which you can edit, if needed). Click **Insert Voucher** and note that the **Credit Meter [Cashable]** changes to \$45.00 when the voucher is redeemed.

Activity 3-4: Exploring G2S Events

Events are one of the most powerful tools in the G2S toolbox. The EGM reports which events it supports, the host can subscribe to whichever events are of interest, and then the EGM will automatically send any of those events whenever the associated activity occurs at the EGM. In this exercise, you will review the events supported by the EGM and the event reports that are generated by the EGM when a supported event occurs (as you'll remember, the RGS subscribes to all supported events in its default subscription).

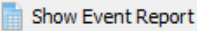
1. Go to the RGS Transcript.
2. Use the Quick Search feature to search for **supportedEvents**.
3. Double-click the `eventHandler.supportedEvents` command to launch the command view.

The screenshot shows a window titled "Transcript Message [ID: 14789] - eventHandler.supportedEvents [G2S_response]". The window displays a list of supported events organized by Device Class, Device ID, and Event Code. The events are listed in a table format with columns for Device Class, Device ID, Event Code, and Event Text.

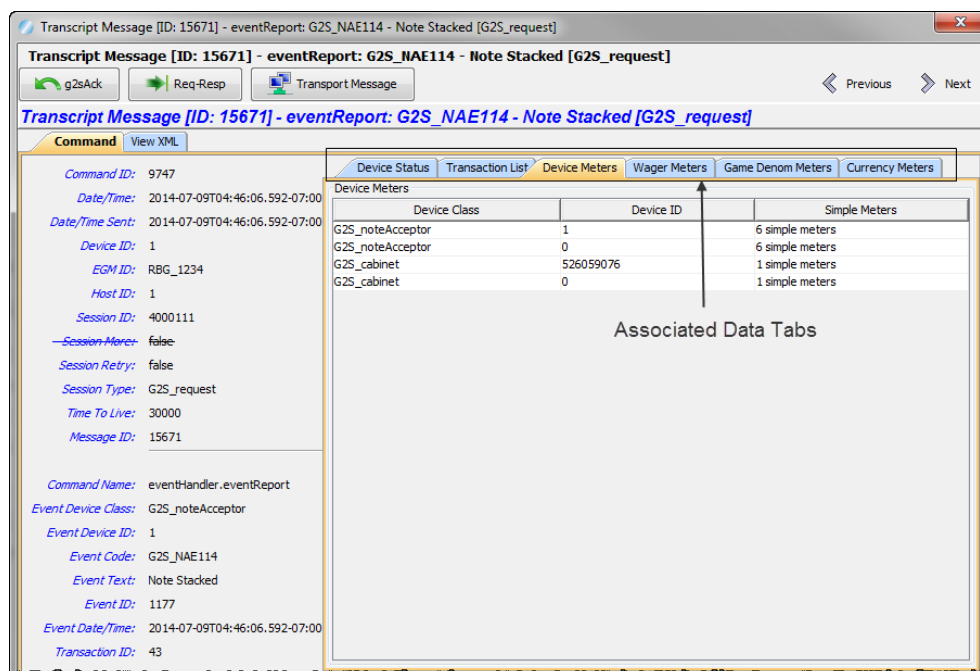
Device Class	Device ID	Event Code	Event Text
G2S_bonus	1	G2S_BNE001	Device Disabled by EGM
G2S_bonus	1	G2S_BNE002	Device Enabled by EGM
G2S_bonus	1	G2S_BNE003	Device Disabled by Host
G2S_bonus	1	G2S_BNE004	Device Enabled by Host
G2S_bonus	1	G2S_BNE005	Device Configuration Changed by Host
G2S_bonus	1	G2S_BNE006	Device Configuration Changed by Operation
G2S_bonus	1	G2S_BNE007	Device Locked by Host
G2S_bonus	1	G2S_BNE008	Device Not Locked by Host
G2S_bonus	1	G2S_BNE101	Bonus Period Started
G2S_bonus	1	G2S_BNE102	Bonus Period Ended
G2S_bonus	1	G2S_BNE103	Bonus Award Pending
G2S_bonus	1	G2S_BNE104	Bonus Award Paid
G2S_bonus	1	G2S_BNE105	Bonus Award Failed
G2S_bonus	1	G2S_BNE106	Bonus Award Canceled
G2S_bonus	1	G2S_BNE107	Bonus Award Acknowledged
G2S_bonus	1	G2S_BNE108	Bonus Host Communications Lost
G2S_bonus	1	G2S_BNE109	Bonus Host Communications Restore
G2S_bonus	1	IGT_BNE001	Bonus Mode Started
G2S_bonus	1	IGT_BNE003	Display Limit Exceeded
G2S_bonus	1	IGT_BNE004	WM Limit Exceeded
G2S_bonus	1	IGT_BNE005	Partial Bonus Award
G2S_cabinet	526059076	G2S_CBE001	EGM Disabled Cabinet
G2S_cabinet	526059076	G2S_CBE002	EGM Enabled Cabinet
G2S_cabinet	526059076	G2S_CBE003	Host Disabled Cabinet
G2S_cabinet	526059076	G2S_CBE004	Host Enabled Cabinet
G2S_cabinet	526059076	G2S_CBE005	Host Changed Cabinet Config

4. Notice that the events that are supported by this EGM are organized by Device Class, Device ID, and then Event Code (event text is included as a handy guide). All event codes that start with "G2S_" are defined by GSA, so a complete description can be found in the **Event Codes** section at the end of the chapter for that class in the G2S protocol document.

5. Scroll down to the **G2S_noteAcceptor** device class and verify that this EGM supports the G2S_NAE114 (Note Stacked) event.
6. Click **OK** to return to the Transcript.
7. Now use the Quick Search feature to search for **eventReport**. All event reports received by the RGS should be displayed. To make it easier to see the contents of each column, right-click on the column header of the Transcript table, and then select **Auto Resize All Columns**.

More Info: If you are interested in viewing only the events in the transcript, you can also select the  button in the top row of the transcript. This causes the transcript to switch to an alternate view that is optimized for events.

8. Scroll down until you find **eventReport: G2S_NAE114 - Note Stacked**, and double-click the row. When the Command View opens, you'll notice a row of "Associated Data" tabs along the right side of the control. G2S describes the data model updates that must occur when each event occurs, and then the EGM sends those updated blocks of data to the host in the `eventReport`.



The screenshot shows the 'Command View' for the eventReport: G2S_NAE114 - Note Stacked. The left pane displays command details, and the right pane shows the 'Device Meters' tab with the following table:

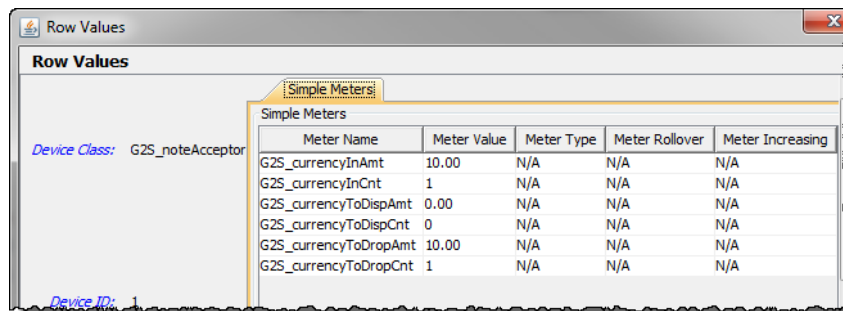
Device Class	Device ID	Simple Meters
G2S_noteAcceptor	1	6 simple meters
G2S_noteAcceptor	0	6 simple meters
G2S_cabinet	526059076	1 simple meters
G2S_cabinet	0	1 simple meters

Below the table, the text 'Associated Data Tabs' is visible.

Selecting the **Device Meters** tab, displays the device meters that were updated when the note was stacked by the EGM (note meters and credit meters increment). You'll notice that there are two instances of the noteAcceptor and cabinet devices (one with Device ID = 0 and then other with a non-zero Device Id). Device ID 0 is used for the "class-level" meters, which are updated when the meters for any device in the class is updated. The non-zero device meters are for the individual device that was updated.

This concept is most relevant in the `gamePlay` class where there are typically many games in the cabinet that are added and removed by commands from the host, so the **class** meters show the totals for the EGM, and the **device** meters (non 0) record the totals for each individual `gamePlay` device.

9. Double-click on the **G2S_noteAcceptor** device to see the 6 meters that are always included for a `noteAcceptor` device. You can also look at the cabinet meters, which are much more extensive.

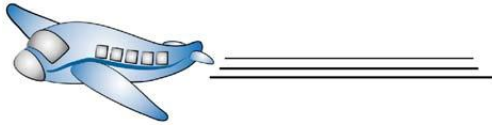


The screenshot shows a window titled "Row Values" with a sub-window titled "Simple Meters". The "Simple Meters" window displays a table with the following data:

Meter Name	Meter Value	Meter Type	Meter Rollover	Meter Increasing
G2S_currencyInAmt	10.00	N/A	N/A	N/A
G2S_currencyInCnt	1	N/A	N/A	N/A
G2S_currencyToDispAmt	0.00	N/A	N/A	N/A
G2S_currencyToDispCnt	0	N/A	N/A	N/A
G2S_currencyToDropAmt	10.00	N/A	N/A	N/A
G2S_currencyToDropCnt	1	N/A	N/A	N/A

10. Click **Back** and then **OK** to return to the Transcript.

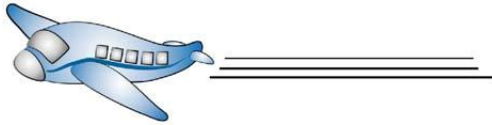
Flying Solo Activity 3-3 – Player Activity at the EGM (RST)



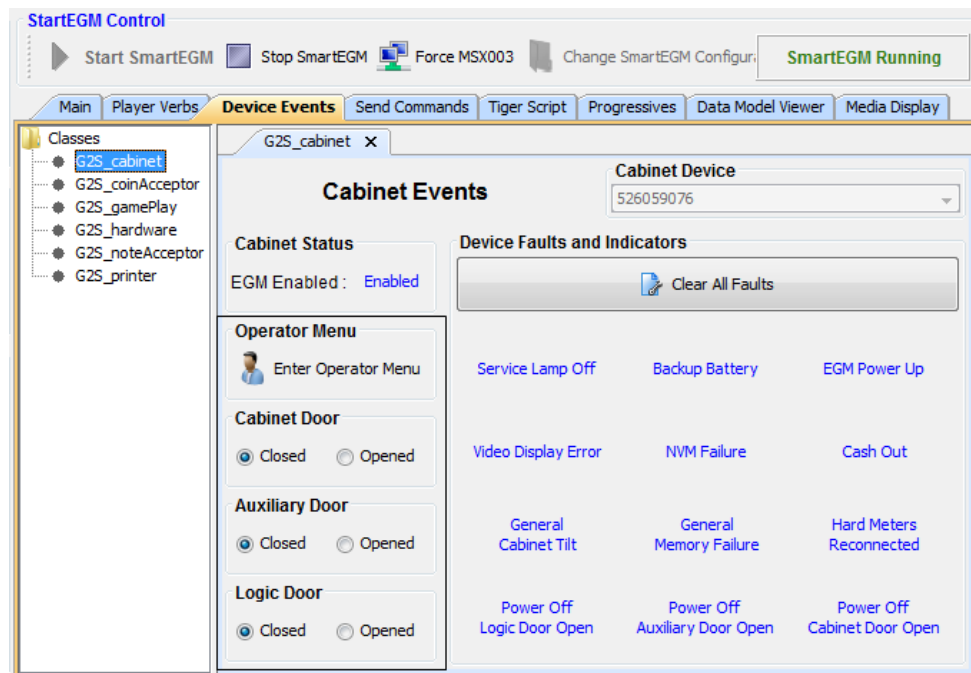
The table below briefly describes each of the General Player Verbs. Explore the G2S commands that are generated when various player actions occur at the EGM.

Player Verb	Description
Cash Out to Handpay	This verb simulates a player pushing the cash out button, and putting the EGM into a handpay situation (aka Cancel Credit).
Create Voucher	This verb causes a cashout of the selected credit meter to a voucher.
Dispense Coin(s)	This verb causes a cashout using coins.
Dispense Note(s)	This verb simulates a cashout to bills (note dispenser)
Insert Coin(s)	This verb simulates the insertion of coin into the EGM's coin acceptor.
Insert ID	This verb simulates the insertion of the player ID (for example, a player card) into the EGM's ID reader.
Insert Note(s)	Simulate the insertion of money into the EGM's note acceptor.
Insert Voucher	This verb simulates the insertion of a voucher (ticket) into the EGM's note acceptor.
Key Off Handpay	This verb simulates the clearing of a handpay condition on the EGM.
Open/Close Doors	This verb simulates the opening or closing of EGM doors.
Perform Coin Drop	Simulates the removal of coin from the EGM.
Perform Note Drop	Simulates the removal of the stacker from the EGM note acceptor.
Play Simple Game	Simulates normal EGM game play.
Play Central Game	Simulates game play using a central determination system. If central determination is not supported by the host, this icon is greyed out.
Play Paytable Game	A more advanced gamePlay mechanism in which you can supply the paytable used by the SmartEGM.
Play Progressive Game	Play a game, and hit a progressive. Select the Progressive level to hit, and then be sure to bet the required number of credits for the proper denomination. When you do, the Play Game button lights up.
Remove ID	Simulates the removal of the player ID from the EGM.
Reset EGM	Resets the cabinet status for lock states that cannot otherwise be cleared through G2S commands or the SmartEGM user interface.

Flying Solo Activity 3-4 – EGM Device Tilts (RST)



Another important group of activity at an EGM is caused when actions occur on the EGM's devices. If you now switch to the Device Events tab and then select the G2S_cabinet device, you'll find a very complete interface that lets you create a wide variety of actions at the EGM, thereby generating lots of events. Since most of these events affect the **status** of the device, try out the device events, and be sure to look at the resulting changes that take place in the Device Status in the eventReport (which you can easily see in the Transcript).



Note: On the Device Events GUI for the cabinet device, most of the items on the left side can be toggled on/off (e.g., **Open** then **Close** one of the doors). On the right side, if you press the **Clear All Faults** button, it clears the previously selected faults, and then returns the EGM to an enabled and playable state.

Entering the Operator Menu exposes a set of Operator Menu Options – each of which create their own G2S events.

Module 4

Configure the Start-Up Algorithm

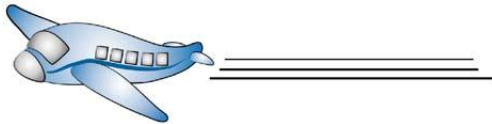
The startup algorithm contains the commands sent to an EGM by RGS when the EGM first comes online (triggered when a `commsOnline` command is received from the EGM).

Normally, RGS sends all commands in this list to the EGM for each device in the `descriptorList` command. If your EGM is having problems with the commands in a particular class, configuring the start-up algorithm provides an easy way to have RGS omit commands from its start-up sequence.

In this module you will learn to:

- start the EGM with communications disabled
- view messages with communications disabled
- enable communications using the RGS Send Command feature

Flying Solo Activity 4-1



Task 1 Launch RGS.

Task 2 Launch RST

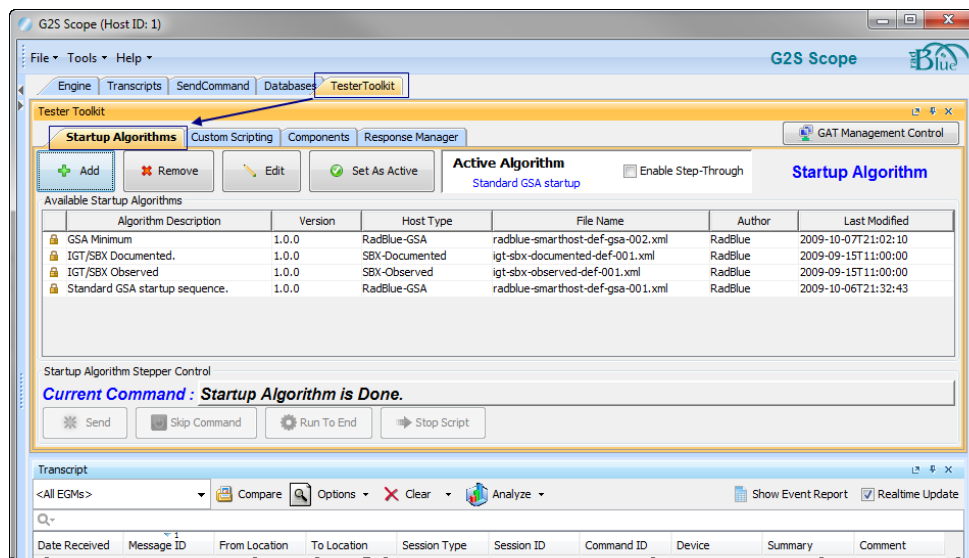
Note Do **not** start the RST SmartEGM engine yet.

Activity 4-1: Configure Startup for Disabled Communications

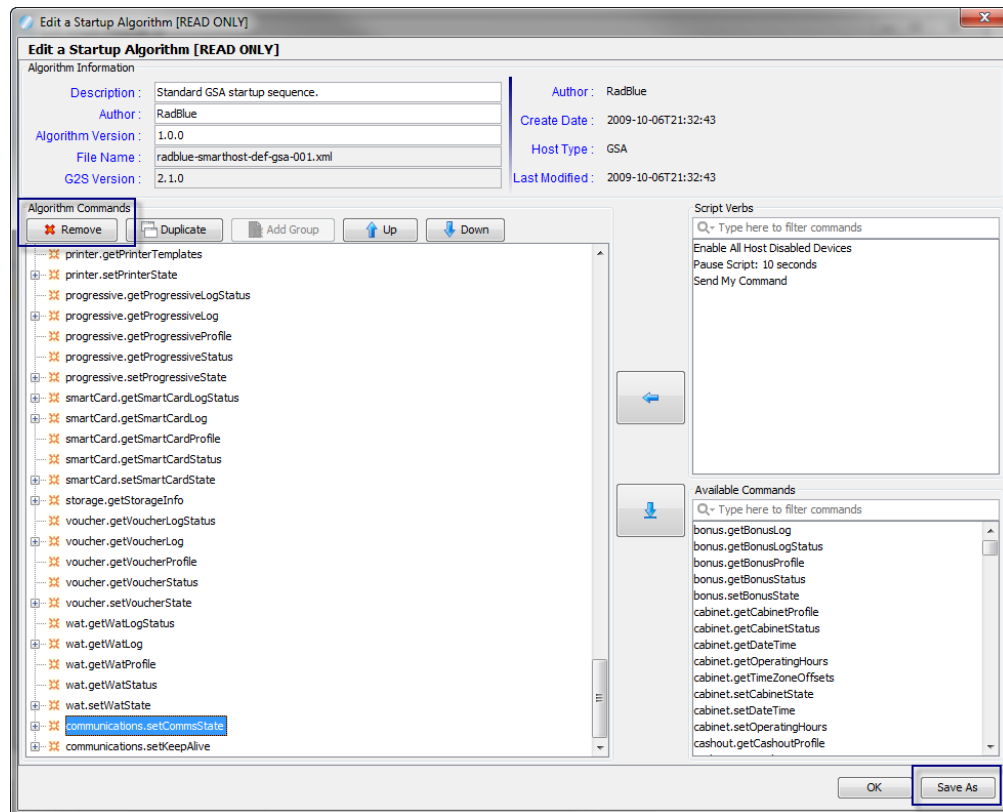
In G2S, the EGM cannot send unsolicited messages to a host until its communication device for that host has been enabled by the host (through a `communications.setCommsState` command). This command is generally sent at the end of the host's startup algorithm when the host has finished querying the EGM and setting event and meter subscriptions.

In this activity, you'll customize the RGS startup algorithm by disabling the `communications.setCommsState` command.

1. Click on the **Tester Toolkit** layout, and then make sure the **Startup Algorithms** tab is selected when the layout opens.

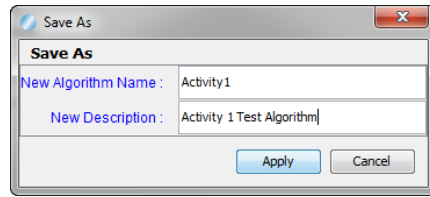


2. Double-click on the "Standard GSA startup sequence" algorithm to launch the editor.
 - a. Notice the Lock symbol in the first column of the Available Startup Algorithm table – this means that file is read-only (but you can save a copy). When you launch the editor you'll also notice the [READ ONLY] warning in the header of the editor (see next page).

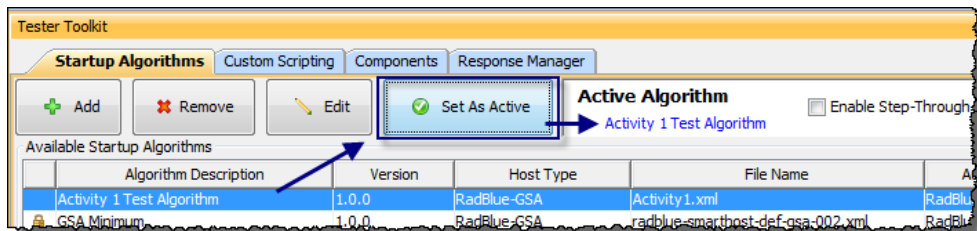


3. In this Editor, the top portion provides three fields that you can edit to personalize this file (the white fields are editable). Since we're going to be saving a copy, take a moment to change the Description, Author, and Version fields.
4. Moving down to the Algorithm Commands section, you see the list of commands that will be sent to the EGM whenever a `commsOnline` command is received to start communications. The commands are only sent if there are one or more devices in that class in the `descriptorList` from the EGM. If there are multiple devices in one class, the commands are sent to each device in the class.
5. Move to the bottom of the list and select the `communications.setCommState` command (we normally don't want the EGM to start chattering until we've completed the configuration process). Press the **Remove** button at the top of the tree to remove that command from the Startup Algorithm. Now, this command will no longer be sent to the EGM during startup.

- Press the **Save As** button and complete the Save As dialogue box as follows:



- At this point, a new “Activity1” algorithm should have been added to the list of **Available Startup Algorithms**. Select the new algorithm, and then press the **Set As Active** button to tell the RGS to start using your new algorithm whenever an EGM connects. A pop-up confirms the action, and the Active Algorithm field is updated.



Activity 4-2: Start EGM with Disabled Communications (RGS)

Now that the RGS is using our modified startup algorithm, start the RST and see what happens.

1. Start the SmartEGM.
2. Look at the Transcript in the RST or the RGS. You'll note that the first `commsOnline` request from the EGM caused a flurry of activity from the RGS – all of which was controlled by the Startup Algorithm file that we edited in the last Activity. Each command in the algorithm is sent and RST sends a corresponding response.

When the start-up algorithm ends, note that no events are generated by the EGM. Instead, RST sends a `commsDisabled` command every 30 seconds, for which RGS sends a `commsDisabledAck` response. “Disabled communications” means the EGM must not generate any requests (other than the `commsDisabled`) until the host turns on the flow by enabling its communications device.

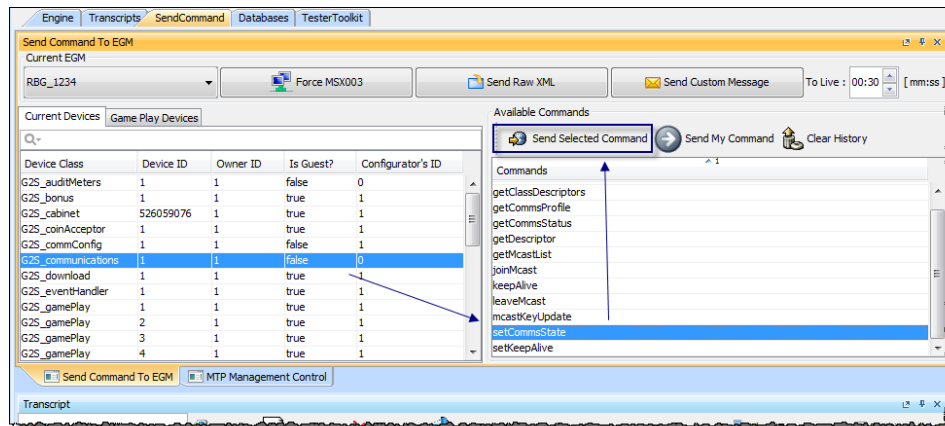
Note The *syncTimer* interval, which is the frequency at which the `commsDisabled` commands are sent as a reminder to the host, can be changed in RGS by going to **Configure > Engine Options > Message > Communications** and then changing the “Communications – Sync Timer” value.

The sync timer interval is sent to the EGM by the RGS in every `commsOnlineAck` and the `commsDisabledAck` responses (take a look!) and is applied by the EGM immediately. This feature lets the host manage G2S traffic when the whole floor goes down and then comes back up again.

Activity 4-3: Enable EGM Communications (RGS)

Now, enable the communications with the EGM by sending a `setCommsState` command.

1. In the RGS, select the **Send Command** layout, and then the **G2S_communications** device.

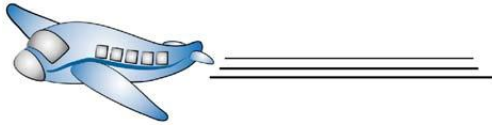


2. Once the communications device is selected, the **Available Commands** list updates to show the G2S commands for a **communications** device. Select the `setCommsState` command, and then press **Send Selected Command**.



3. Click in the Value column for the **Enable** field, and select **True**.
4. Click **Send Command**.
5. Look at the Transcript. Now that the communications device is enabled, normal messaging begins, so all subscribed events are sent and the `commsDisabled` command is no longer generated. Life is good.

Flying Solo Activity 4-2



- Task 1** Use the Send Command and Configure option to enable/disable various G2S devices. Use the Transcript to see the effect of each enable/disable action.
- Task 2** Stop the SmartEGM in the RST, and then close both tools.

Time for another break!

Module 5

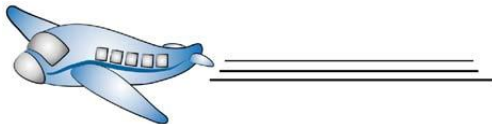
Adding the Protocol Analyzer (RPA)

Now that you know how to get RST and RGS communicating, let's insert RPA between those two endpoints.

In this module you will learn to:

- navigate the RPA user interface.
- configure RPA to communicate with RST and RGS.

Flying Solo Activity 5-1



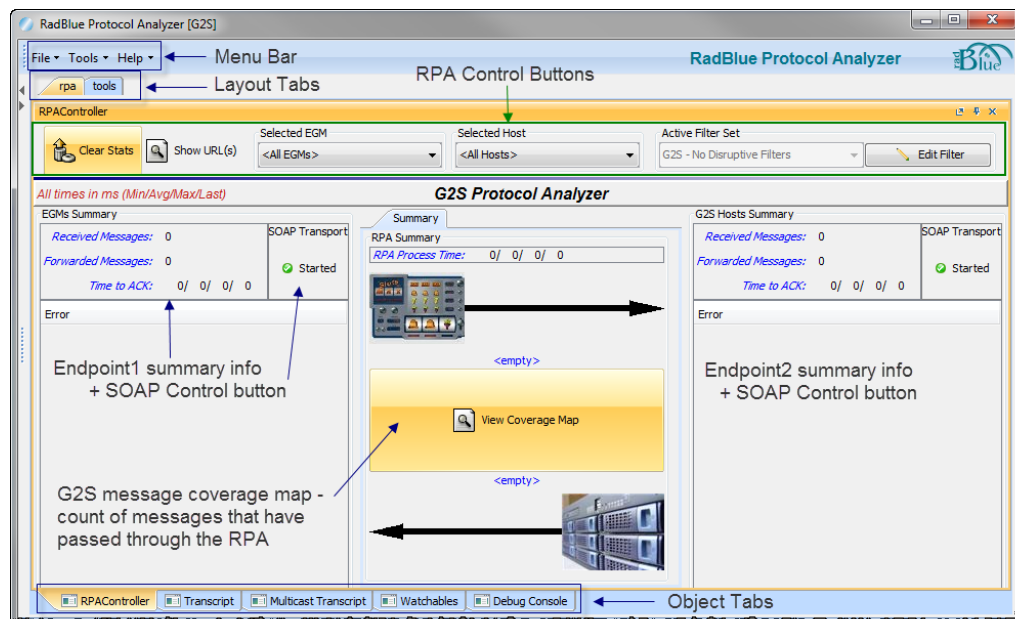
Task 1 Launch RPA by double-clicking the RadBlue Protocol Analyzer icon.

When the RPA starts, it automatically launches its web-server, and then waits for connections.

Activity 5-1: A First Look at the RPA

Take a few moments to familiarize yourself with the RPA interface. You'll find that the general layout is similar to RST and RGS, with a familiar Menu Bar at the top, followed by a couple of Layout Tabs (rpa and tools), and then a series of object tabs (this time with the tab buttons on the bottom) are used to divide up the functionality. Your old friend, the Transcript, is back as well.

The RPA Controller layout changes slightly depending on whether or not you are using disruptive filters. For the purposes of this guide, we're going to assume that you are not using disruptive filters, so your instance of RPA should look like the screenshot, below. For more information on disruptive filters, see the [RPA User Guide](#).



There are three main sections on the RPA layout:

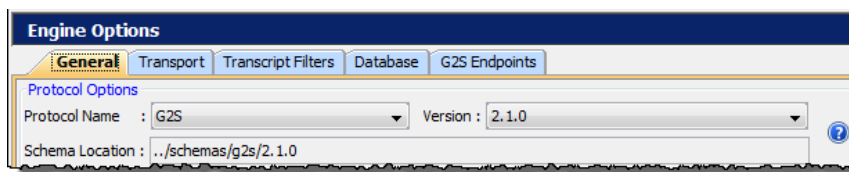
- **RPA Control Buttons** – A set of controls that let you clear statistics, view the RPA URLs to be used by the endpoints, and select a specific host or EGM for the statistics (RPA supports up to 5 G2S EGMs communicating to 5 G2S Hosts)
- **Endpoint Summary Areas** - These sections on either side of the object display show statistics for commands sent by that end-point (EGMs or Hosts for G2S) as well as any identifying any messages with errors. A SOAP Transport button lets you create SOAP faults by stopping the RPA web-service used by that endpoint.
- **RPA Summary** - This section displays the most recent command from each endpoint. Click the View Coverage Map to see a list of all G2S commands and events and the number of times RPA has received each.

Activity 5-2: Configuring the RPA – a Quick Overview

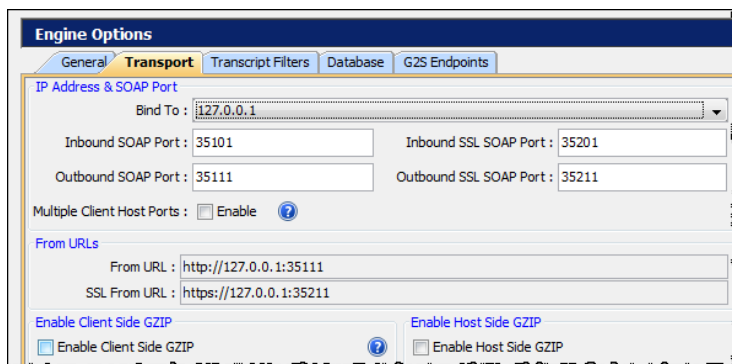
Hmmm – it sounds daunting, but though the RPA is really powerful, it is quite easy to use. In this activity, we'll hit on the high-points of configuring the **Engine Options** in the RPA.

Start by opening the Engine Options Configuration Utility (**Tools > Configure > Engine Options**), then

1. Select the **General Tab**. On this tab you can select the **Protocol to Use (G2S or S2S)**, and then a version of that protocol to use. For G2S, the tools will always default to 2.1.0, a special version of the schema we use in our tools that we've tested to ensure it's compatible with older versions of the protocol. If you want to use a special schema containing your secret sauce, just look to the [RPA User Guide](#) for easy to follow procedures on how to add your schema to the RPA's schema directory.



2. Next, select the **Transport** tab. On this tab, you can select which IP address (network interface) that you would like the RPA to “**bind to**” (connect to). For this exercise, just pick localhost (127.0.0.1) as all of our communications will be running on the same PC, without going out to the network.

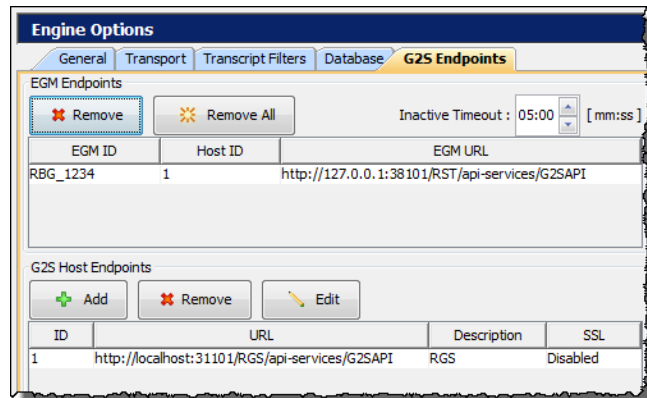


- a. **SOAP Ports** – this is the port number (like a street address) that the tool listens to on that network connection. The rule is that each instance of the tool **MUST** use a unique port number (just leave these alone for now).
- b. **Multiple Client Host Ports** – some EGMs require that every host have a unique URL. If you select this checkbox, the RPA will use a range of port numbers to make the URLs unique for each of the hosts it routes messages to.

- c. **From URLs** – In these fields, the RPA shows the URLs it will be using (you can also see these on the main RPA Controller screen using the **Show URLs** button).
 - d. **Will the end-points need GZIP?** – GZIP is an optional compression algorithm for G2S. This option configures how the RPA responds when asked if it supports GZIP.
3. Select the **G2S Endpoints** tab. To keep track of all of the connections (5 EGMs to 5 Hosts), the RPA uses a **G2S Endpoint Map**.

In this control, the **EGM Endpoints** table is automatically populated as each EGM connects to the RPA. The RPA only supports 5 EGMs in the table, so options are included to remove one or all of the historic EGMs.

The data in the **G2S Host Endpoints** table is manually entered as the RPA needs to know the unique hostId for each G2S host, how to get to that host (the URL), and whether it is using GZIP and/or SSL.

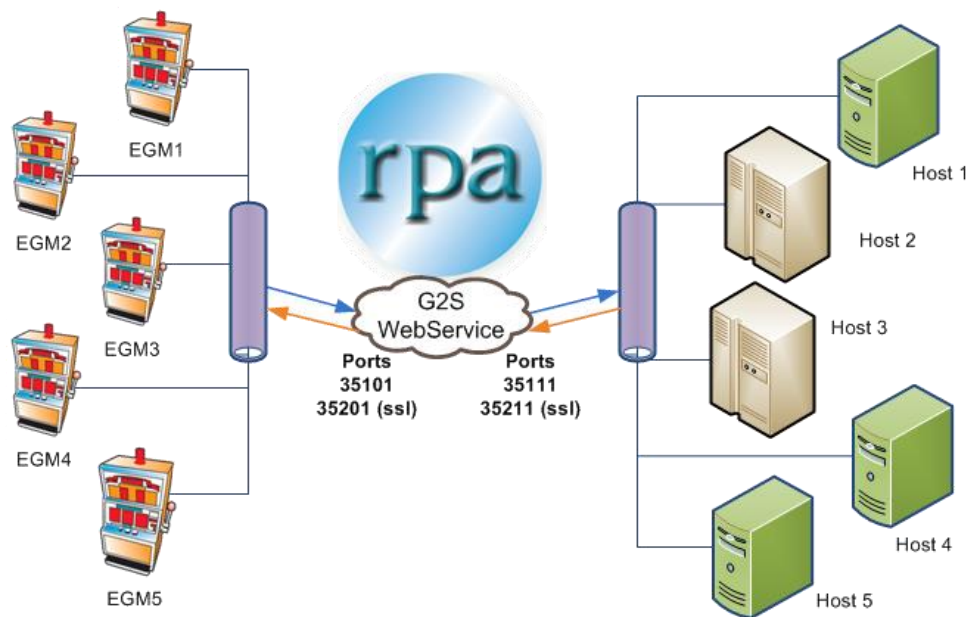


For our exercises, just review the information that is presented (no changes are needed). When first installed, the RPA is designed to work with the RST and the RGS in localhost mode...

4. If you’ve made any changes, press **Cancel** to discard them, or **Apply** then **OK** to accept the changes. If you’ve changed the URL being used by the RPA, you’ll be prompted to restart the tool so it can use the new values.

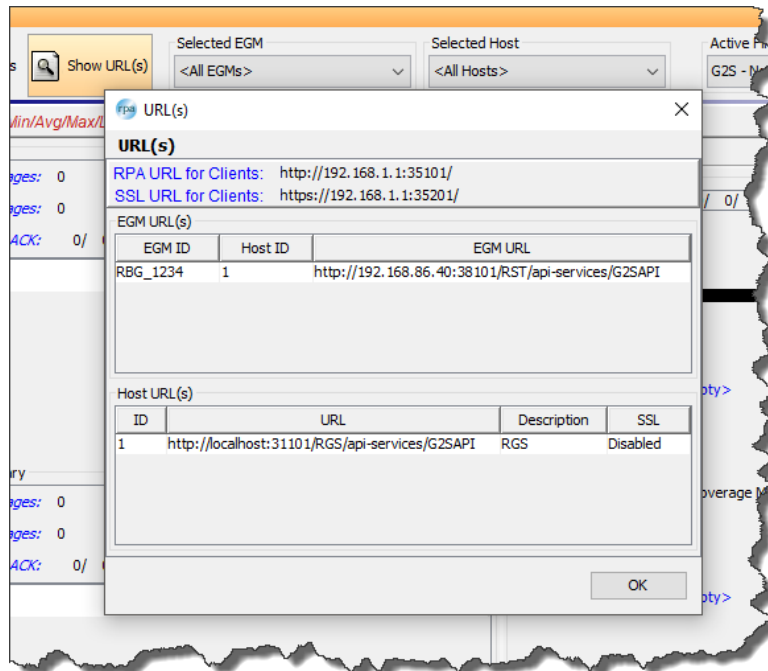
Activity 5-3: How to get RPA in the middle?

Up until now, the RST has been connecting directly to the webserver in the RGS, then RGS connects back to the webserver in the RST, and then G2S happens. Our goal now is to have this same G2S conversation going on between RST and RGS, but to have RPA in the middle. To do this, we just have to tell the EGM (RST) to contact the RPA, rather than the RGS, and then tell the RPA how to get to the host (RGS – which we’ve seen it knows how to do by default). It’s actually a bit easier than it appears....



Change the RST's SmartEGM Configuration File – To connect the RST to RPA you will need to modify the host URL to specify RPA URL.

1. Click the **Main** tab on the RPA, if it is not already selected.
2. Click the show URL(s) button. This will show the URLs that you need to enter into your slot machines and in our case the RST.



RPA URL for Clients is the URL that enter in your EGM for NON-SSL traffic (unencrypted). **SSL URL for Client** is the URL that you enter for encrypted communications.

3. In the RST, click the **Main** tab on the **SmartEGM** layout, if it is not already selected.
4. Select **Host ID 1** in the **Hosts** tab and then click **Edit Hosts**.
5. Set the Description to The RPA and the Host URL to the RPA URL for Client. For student editions you will need to use the SSL URL. Note, the IP address will be

different on your computer.

6. Click **Save**.
7. Click **Start SmartEGM**.

Activity 5-4: View G2S Data in the RPA

1. Look at the three summary sections in the RPA to view messages flowing through RPA.
 - a. The EGMs and G2S Hosts Summary sections show you the number of messages received from (and sent to) the EGMs and hosts. The timing statistics show you how long it took for the messages sent to that end point to be acknowledged (by a g2sAck). Times are in milliseconds, and all show the **Minimum / Average / Maximum / and Last** times.
 - b. The Summary section in the middle of the screen shows the RPA process time statistics, as well as the most recent G2S messages being handled by RPA (from each end-point).

Note: If you change the Selected EGM or Selected Host in the RPA Controller section of the screen, all statistics are changed so they reference only the messages for the selected end-point (EGM and/or Host)

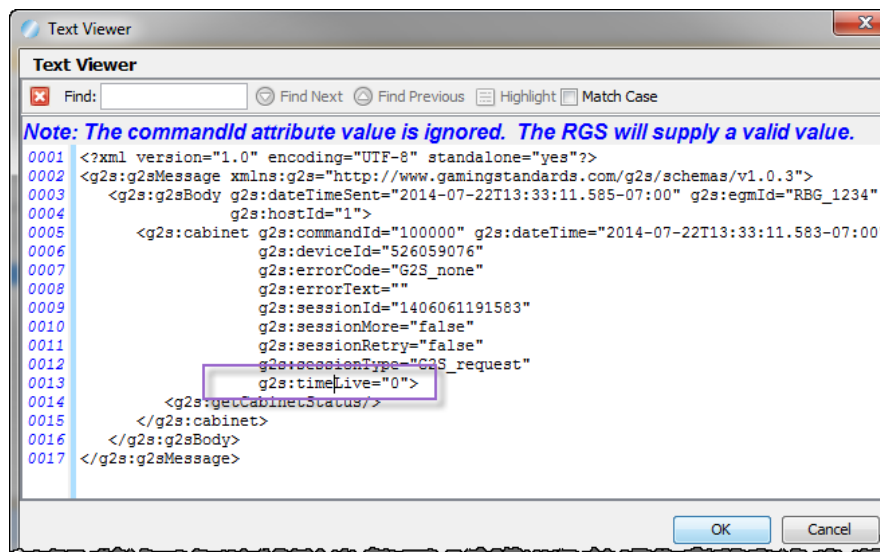
2. Click **View Coverage Map** to see the number of messages received for each message and event. Notice that commands and events that have not been received by RPA appear in red.
3. Use the Quick Search feature in the **Coverage Map** to search for the `setCabinetState` command.
4. Now search for `eventReport`, and use the scrollbar to see the events that have been sent between the two endpoints. Since there are quite a few events, you may want to

click on the Count column header to sort the list from the largest count to smallest. That way, all of the `eventReport`'s with non-zero counters move to the top of the list.

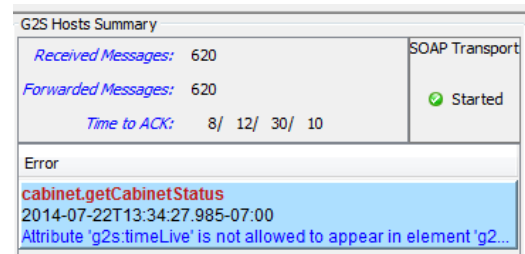
Activity 5-5: Invalid G2S Messages (RGS)

In this section, we'll send an invalid G2S message from the RGS and then explore the tools in the RPA that let you easily research bad messages – who caused the problem, and what exactly is wrong with the message – making it very easy to accurately report a problem, and (hopefully) get it fixed.

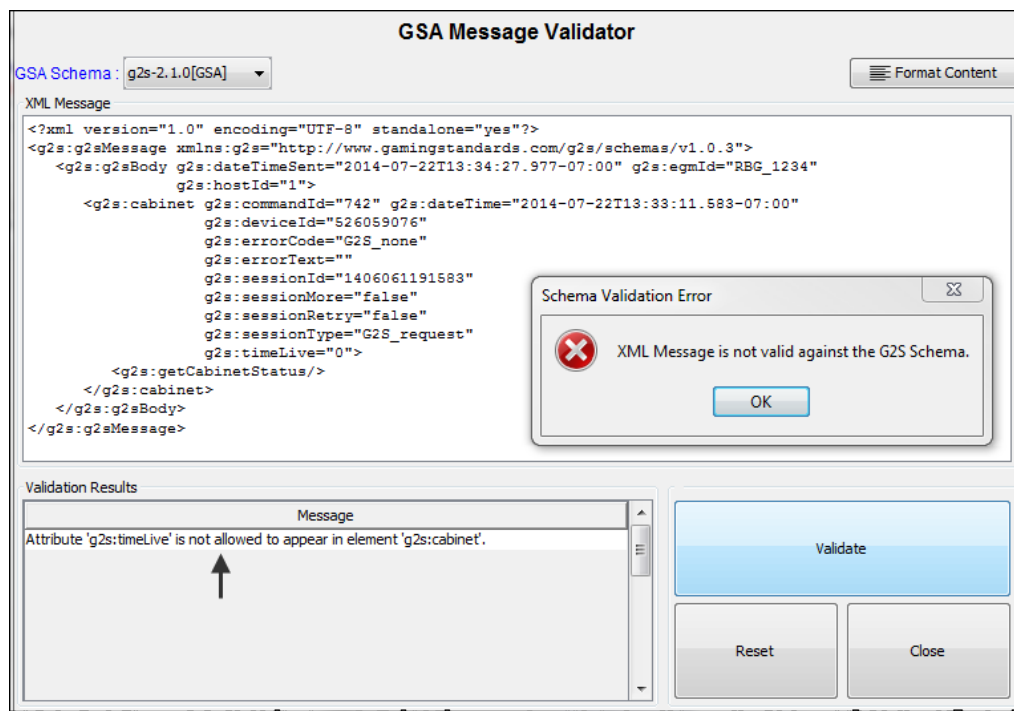
1. In the RGS, select the **SendCommand** layout, and then select the G2S_cabinet device (to send a command to the cabinet device in the EGM).
2. Double-click `getCabinetStatus` to launch the Send Command screen for that command.
3. Select **Custom Send** to modify the command just before it is sent by the RGS. At this point, the command is editable, so select the `g2s:timeToLive` attribute and remove the "To" so you end up with `g2s:timeLive="0"`. Then press **OK** to send the command.



4. In the RPA, in the **Error** section under the **G2S Host Summary**, you'll notice that an error has been reported for the `getCabinetStatus` command that was sent by the host. The summary indicates that an unknown attribute (`g2s:timeLive`) was found in the message.
5. If you double-click the error message, an **Error Browser** screen is launched that shows more complete details of this request (including the complete message [View XML] and the response that was returned by the EGM (in this case a **G2S_MSX004** error which signals that the EGM also detected the error in the message).



6. Select **View XML**, and then copy the message (highlight the whole message, and then press Ctrl-C to copy it to the clipboard).
7. Now, close the **XML Payload** browser and the **Error Browser**, and open the **GSA Message Validator** (under **Tools** on the Menu Bar).
8. In the XML Message section of the **GSA Message Validator** screen, highlight the **"Replace XML content here."** text, and then paste the XML from the Error Browser, then press the **Validate** button.



9. Using this method, we can manually check any message against the specified schema (sort of like spell-checking a document). In this case, the pop-up window shows the pasted XML message is not valid against the selected G2S Schema (g2s-2.1.0(GSA)), and the Validation Results window indicates the exact problem with the message. With this type of information, anyone can find errors and report complete details of the error so it can be easily fixed.

Time for another well-deserved break. While you're relaxing, try to send a few more errors from the host using the Custom Send strategy we used earlier in this section, or try validating some messages from the transcript...

Module 6

Advanced Skills

In this module, you will learn to:

- configure RST and RGS to run on two different computers
- add RPA between RST and RGS running on different computers
- run the Advanced Transcript Analysis report – a powerful way to find a needle in a haystack (actually, an error in the transcript)
- explore changes in the EGM's Data Model
- download and run mediaDisplay content in the RST (Player User Interface)
- explore the RGS Custom Scripting module to do advanced automated testing of an EGM

As this module represents more advanced exercises for the tools, we assume that you have already completed the activities in the prior chapters. As a result, less explanation of the basic operations will be provided as we tackle more interesting concepts.

As always, if you have any questions, we're always around. Just e-mail us at support@radblue.com to start a conversation about our tools, or the G2S protocol.

Good luck!

Activity 6-1: Run RST and RGS on Two Different Computers

Once you have mastered running RST and RGS on the same computer, you may want to run them on two different computers to get a better understanding of network communications. Using two different computers (that is – two different nodes on the network) provides a more realistic messaging simulation.

1. Install the RST on Computer 1 and the RGS on Computer 2. Make sure you have the appropriate licenses for each computer, as each license is tied to a specific MAC address. If you need additional student licenses, just send us the MAC address of the additional machines in a request to license@radblue.com.
2. Install the RPA on either of the computers (we'll use the RPA again in a later activity).
3. Start the RGS and the RST, **but do not start the SmartEGM** (just yet). Make sure the RST is using the standard configuration file (not one for the RPA).

When an EGM connects to a host, it needs 1) to know the network location (URL) of the host application's webserver, and 2) it has to tell the host how to connect to the EGM's webserver. The EGM location is provided in the *egmLocation* attribute of the `commsOnline` command (the first G2S command sent to the host by the EGM, indicating "I want to connect to you. Here's my EGM ID and my Network Location so you can get back to me."), as seen in the following snippet from a `commsOnline` command.

```
<g2s:commsOnline xmlns:g2s1="http://www.gamingstandards.com/g2s/schemas/v1.0.3/Ext1"
  xmlns:rbg="http://www.radblue.com/gsa/g2s/extensions/1.0.0"
  g2s1:deviceAccessChanged="false"
  g2s1:deviceStateChanged="false"
  g2s1:negotiateNamespaces="true"
  g2s:deviceChanged="false"
  g2s:deviceReset="false"
  g2s:egmLocation="http://172.16.53.52:38101/RST/api-services/G2SAPI"
```

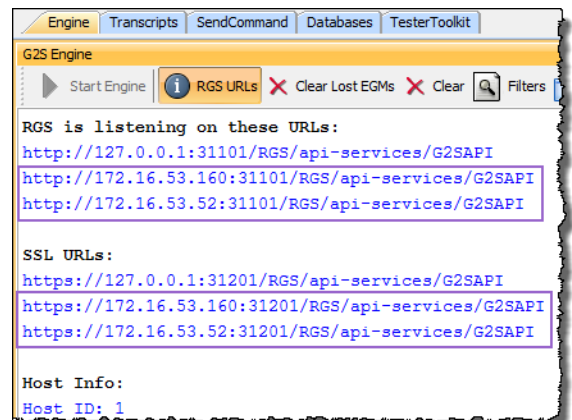
4. First, we need to change the URL of **host 1**, so it references the RGS's network address, rather than using localhost (the default).

In the RGS, click **RGS URLs** on the Engine layout, to see the URLs being used by the RGS.

Note: 127.0.0.1 is the IP address for **localhost**

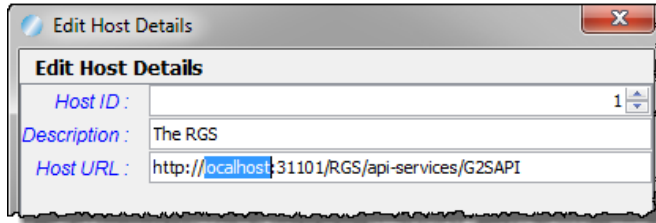
If you are using a standard license, use any of the http: URLs (one for each network card).

If you are using a student license, select any SSL URL (G2S communications are encrypted with a student license).



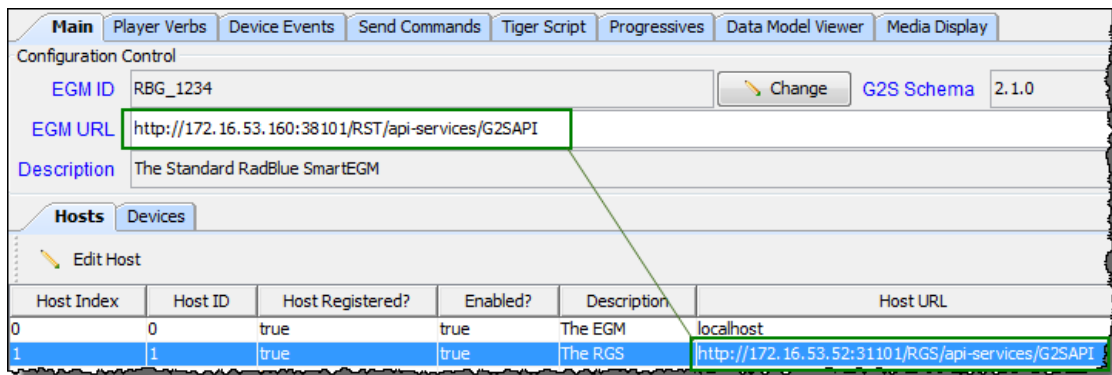
- On the RST’s SmartEGM layout, select **Host Index 1** (the RGS), then click **Edit Host** to launch the **Edit Host Details** screen. To make things easy, just change “localhost:” to one of the IP addresses used by the RGS, then click **Save** to save the new URL for that host.

Note: Don’t change the port number or the rest of the path or you’ll connect the web-server, but the message won’t get to the RGS.



Now that the RST knows how to reach the host, we need to tell the RST to listen on the Network (rather than localhost) and then we need to make sure the RGS is told how to get back to the EGM.

- First, tell the RST to “**Bind To:**” (listen on) an IP address on the network. On the Menu Bar, select **Tools > Configure > Engine Options > Transport** to get to the configuration screen for the network parameters. At the top of the screen, click on the **Bind To:** dropdown list and select an IP address other than 127.0.0.1 (localhost). Click **OK** to accept the changes, after which you are prompted to restart the RST so it can set up a web service that uses the new IP address.
- Restart the RST
- When the RST starts up, on the SmartEGM layout, make sure the EGM URL and host 1 both show the network IP address (not localhost or 127.0.0.1).



- Start the SmartEGM engine. Look at the `commsOnline` to see the new *egmLocation*. If all changes were made correctly, you should now see communications flowing between RST and RGS. Review the Debug Log in each tool for any errors

Activity 6-2: Add RPA between RST and RGS (again)

Whether RPA is installed on the same computer as RGS or the RST, the configuration procedure is the same: configure the RST to use the RPA's network IP address and then configure RPA to use the network IP address for the RGS.

1. Stop the SmartEGM in the RST if it is currently running.
2. Start the RPA on either computer. As with the RST, we need to change the IP address to which the RPA is binding (the RGS automatically binds to all addresses).
3. In the RPA, go to **Tools > Configure > Engine Options > Transport**, click the **Bind To:** dropdown, and then select an IP address on the external network interface (not 127.0.0.1).

Click the **G2S Endpoints** table (still in **Engine Options** configuration), as we need to update the IP address for the RGS (G2S Host 1) so the RPA knows how to get there.

In the lower portion of the screen, double-click Host 1 (RGS) to launch the **G2S Host Data Editor**. Change **localhost** in the Host URL to match one of the IP addresses on which the RGS is listening (see Step 4 in Activity 6-1 if you need a reminder).

4. Click **OK** to accept the changes, after which you are prompted to restart the RPA so it can set up a web service using the new IP address.
5. Restart the RPA
6. On the RPA, click on **Show URL(s)** to show the URL(s) table. At the top of that screen, you'll see the **RPA URL for Clients** and **SSL URLs for Clients**. As both use the same IP address (but different port numbers), just copy the IP address as we'll need to tell the RST how to get to this host.

Now – to tell the EGM (RST) how to get to the RPA:

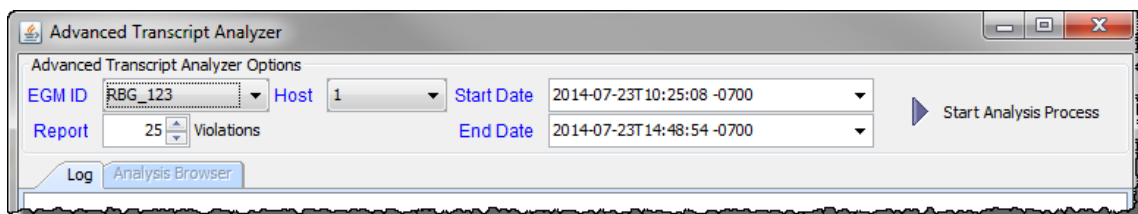
7. Click the **Main** tab on the **SmartEGM** layout (RST), if it is not already selected, then click **Change SmartEGM Configuration**.
8. Select **smartegm-config-gsa-rpa.xml** to highlight it, or if you are using a student license, select **smartegm-config-gsa-student-edition-rpa.xml**, and click **Open**.
9. On the RST's SmartEGM layout, select **Host Index 1** (which should be the **RPA**), then click **Edit Host** to launch the **Edit Host Details** screen. Change "localhost:" to the IP address being used by the RPA, then click **Save**.
10. Start the SmartEGM engine in the RST, and verify that the G2S messages between the RST and RGS are flowing through the RPA in both directions. If not, check all IP address settings and try again.

Activity 6-3: The Advanced Transcript Analyzer (ATA)

The Advanced Transcript Analyzer lets you easily verify that G2S commands being sent by the EGM are **syntactically** and **semantically** valid. **Syntax** checking occurs when the message is validated against the appropriate schema to ensure that all rules defined in the schema are followed. **Semantic** checking is performed using an extensive set of rules that are defined in the protocol document, and have been coded in the Advanced Transcript Analyzer (ATA) engine. These rules are then automatically applied against all G2S messages in the transcript to make sure they follow the rules of the protocol.

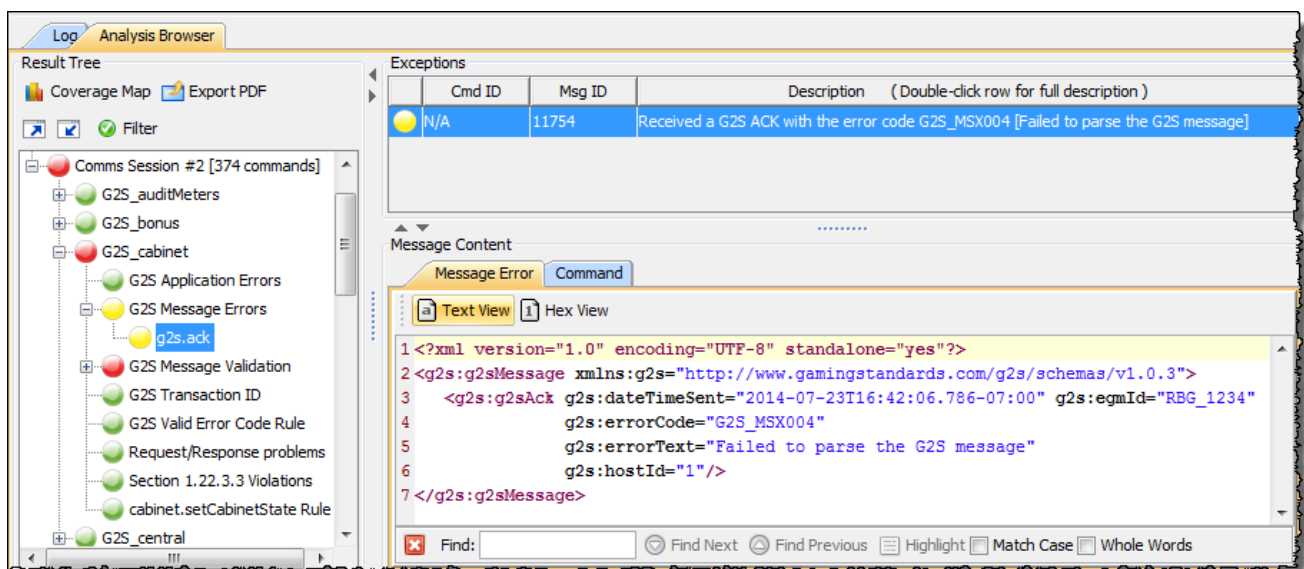
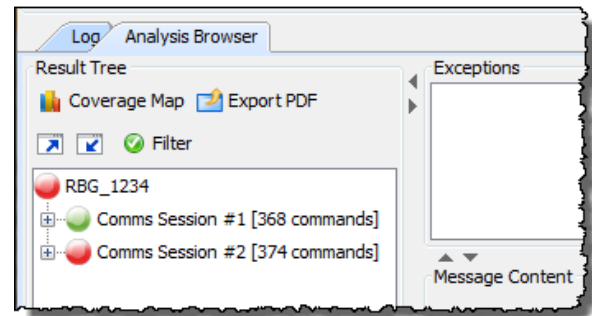
Visual cues let you easily discern which events and commands have errors. You can then quickly drill down to the message content level to view the issue, and related commands where appropriate. In this exercise we'll generate some message errors and will then use the ATA to find those errors in the transcript.

1. Start the RST and RGS if they are not already running (the RPA is not needed for this step, but can be running in the middle, if desired). Start the SmartEGM in the RST to generate some traffic in the transcript.
2. Insert a \$10 Note in the RST to put money on the credit meter (**Player Verbs – Insert Note** on the **SmartEGM** layout in the RST).
3. Stop the SmartEGM and then restart it again. This will cause some application errors when commands are attempted by the host and there is money on the credit meter.
4. On the RGS, select the **SendCommand** layout, then the **G2S_cabinet** device, then the **getCabinetStatus** command. Click on **Custom Send**, then change the value of *g2s:sessionType* (line 12) from "**G2S_request**" to "**G2S_reques**t" (remove the "u" in **request**), and then press **OK** to send the command. Notice the MSX004 error from the EGM.
5. An instance of the Transcript object is on the bottom half of the SendCommand layout (so you easily can see messages as you send them). In the top border of that Transcript object, click **Analyze**, then **Advanced**.

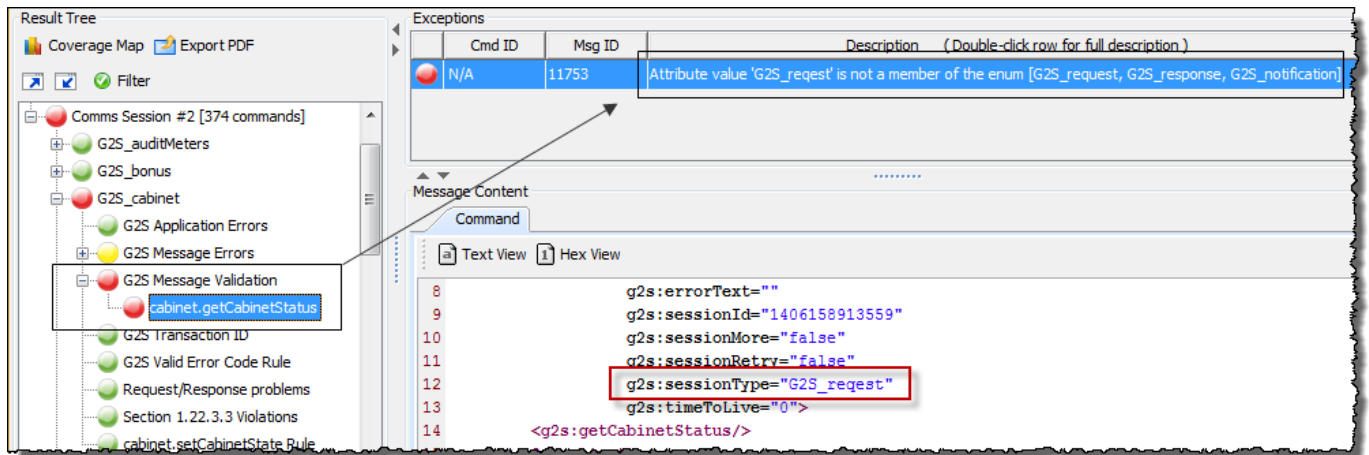


The ATA Options screen appears, which allows you to select for a particular host or EGM, or a particular time range. For this example, just click **Start Analysis Process**. Progress updates will appear in the log, and when the analysis is completed, the **Analysis Browser** displays with the results.

6. On the left side of the Analysis Browser is a **Result Tree** that has one or more **Comms Session** nodes (we define a “Comms Session” as a set of G2S commands, starting with a `commsOnline`, followed by a `descriptorList`, and ending with a `commsClosing` or another `commsOnline`). If a node is not Green, it contains one or more errors. Click to expand a non-green Node.
7. Doing so reveals a new set of nodes – one for each G2S class in the G2S schema. If a class is not present, or no errors are encountered, the class node is Green. If there are WARNINGS, it is Yellow, and if there are ERRORS, it is Red.



8. Selecting a leaf node (a node that can't be expanded) causes the panels on the right to be populated.
 - a. The Exceptions table (at the top) provides a high-level summary of the error (including the message Id so you can easily find that message in the transcript to see the surrounding context of the message),
 - b. The Message Content section at the bottom shows the actual message (here it's the error message), and the Command that caused (or is related) to this message.
 - c. This node is a WARNING, since we can't tell how serious a message error might be – just letting you know it's there so you can check it out.

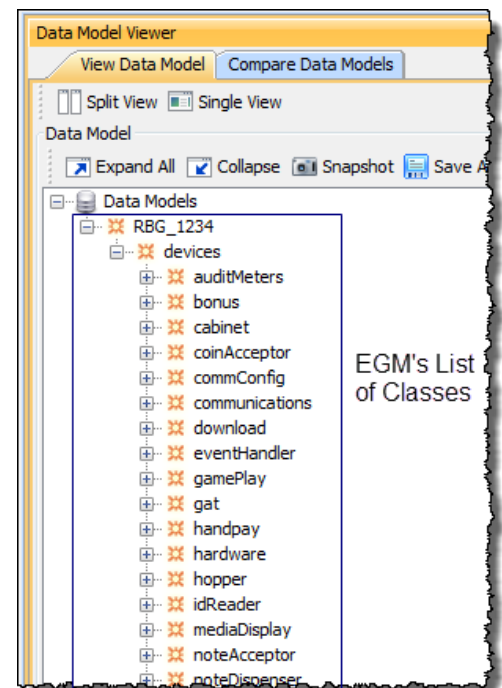


9. Selecting the more serious **G2S Message Validation** node, and then the `cabinet.getCabinetStatus` command underneath it, we are now provided with more complete details of the problem (the value “G2S_request” is not one of the allowed choices for this attribute), and a copy of the command in error.
10. To easily create a report of these errors, just select the **Export PDF** option, which creates a PDF format report of the errors.
 If desired, you can filter the tree before creating the report (via the **Filter** button).
 When you create the report, you are given the option of Summary (report one instance of each error per class), or Complete (report all errors in all classes), and you can specify the target location for the report. Give it a try!
 For more information on the Advanced Transcript Analyzer function, take a look at the chapter devoted to this topic in the [RGS User Guide](#).
11. At this point, close the ATA, the RGS, and the RST and take a short break.

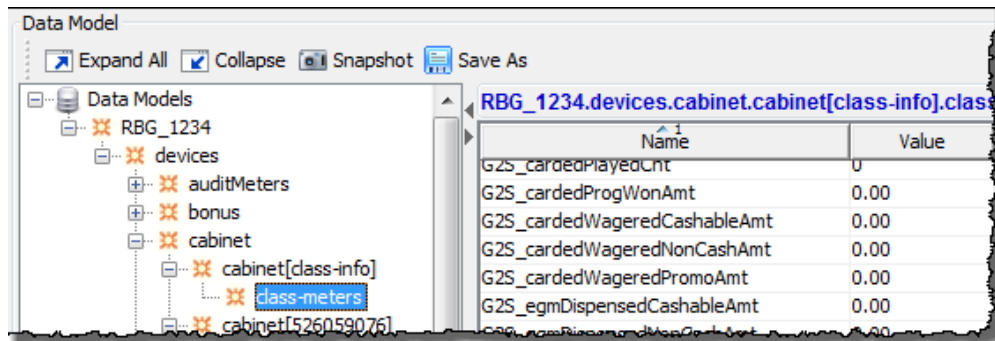
Activity 6-4: Exploring the EGM Data Model (RGS)

One of the most powerful features of G2S is the ability for a host to view everything in the EGM's *data model*, which is the collection of G2S data maintained by the EGM, including current parameter settings, status information, and meter values for each of the G2S devices. In addition to being able to access any of these data elements at any time (via “**get**” commands), the functionality of the **eventHandler** class provides a mechanism by which the host's view of an EGM's data model can be updated in real-time. In this activity, we'll explore the EGM data model using the powerful *data model viewer* (DMV) in the RGS.

1. Restart the RST and the RGS, but don't start the SmartEGM in the RST, just yet. For this activity, we want to start with a clean slate in the RGS.
2. In the RGS, select the Data Model Viewer tab on the Databases layout. You should see the start of a tree, with only a **Data Models** container as the root node, but nothing else. This shows that the RGS has no information about any EGMs before they connect.
3. Make sure the RST is pointing at the RGS (check the **Hosts** table, and reload the non-RPA configuration file if necessary). Start the SmartEGM.
4. Back in the RGS, you'll notice that a + symbol has appeared before the **Data Models** node. Clicking on that + symbol opens the tree to the connected EGMs. Clicking on the + symbol in front of our EGM (egmId=RBG_1234) reveals a devices node, and then another click on the devices node shows all of the classes supported by the EGM (at this point, your screen should be similar to the snippet on the right side of this page).

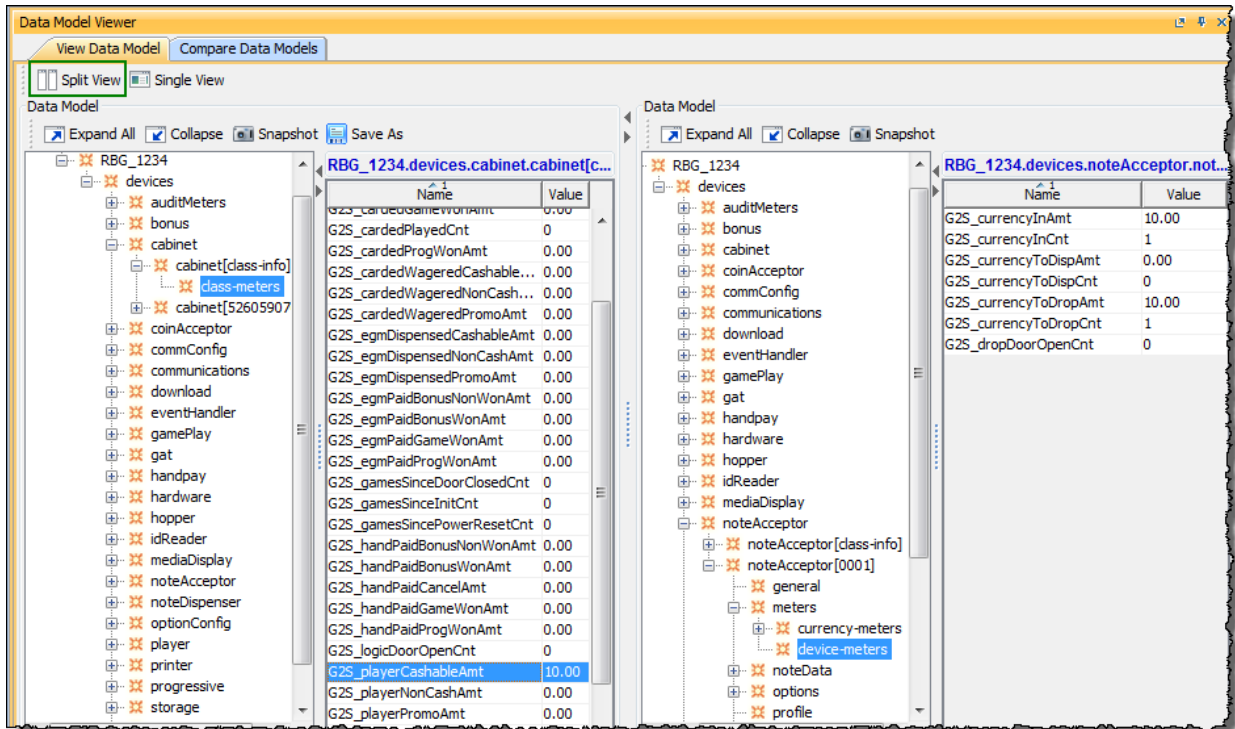


- Now, select **cabinet**, then **cabinet[class-info]**, then click on the **class-meters** “leaf node” (the last node in the branch that can no longer be expanded – there’s no + symbol in front of it). The table on the right side of the control now shows the Name-Value pairs for each of the cabinet meters – as they are in the EGM right now. During the Startup Algorithm (remember that list of commands sent by the RGS when a commsOnline command comes in?), the RGS sends all of the G2S “get” commands to read all of the EGM data that’s available through G2S, and then populates its internal data model for that EGM. This control lets you explore that data.



In addition to being able to view changes to the EGM’s data in real-time, the RGS can also create a snapshot of the entire data model, and can then compare any two snapshots to see the changes that occurred during any time period. Let’s create a starting snapshot that we’ll use later to try out the compare feature.

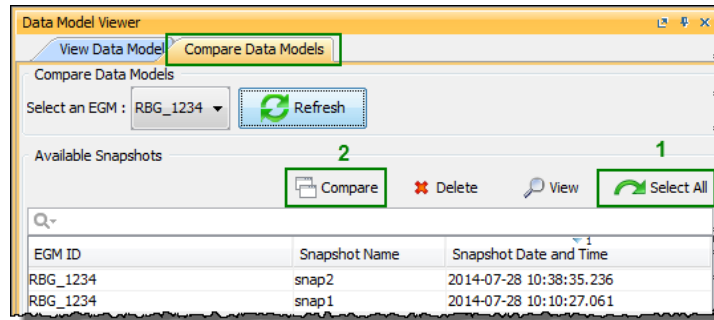
- Click on the Snapshot icon at the top of the data models tree. A new screen will pop up, prompting you to select an EGM (select **RBG_1234**), and also to provide a name for the new snapshot (let’s use “**snap1**” so we’re all on the same page). In a moment or two, a new screen is displayed acknowledging the successful operation. We’ll come back to this in a few moments.
- Sometimes, it’s handy to watch meter movements in two devices at the same time, so the DMV provides a **Split View** in addition to a **Single View** (the default). Select **Split View** at the top of the object window and you’ll see that a second Data Model tree is now displayed. Since you can separately navigate the tree in each view, try to display the cabinet class-meters (specifically, **G2S_playerCashableAmt**) in the left view and the **noteAcceptor** device meters in the right view.
- Your display should now look something like the example on the top of the next page:



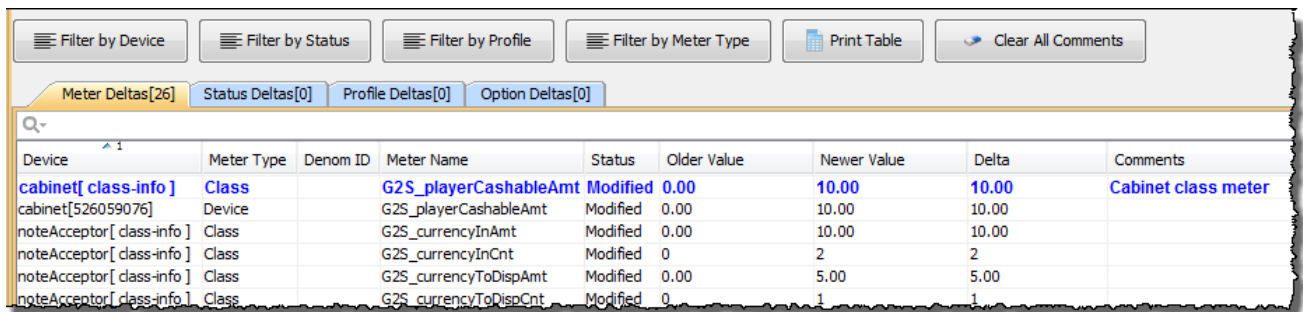
- Now that we can watch the EGM credit meter and the noteIn meters at the same time, try inserting a \$5 note in the RST (**Player Verbs > Insert Note** in case you need a reminder). As soon as the note is accepted by the EGM, the credit meter on the EGM increments by \$5 and the corresponding meters in the DMV are magically updated.

To see this meter update in the transcript, switch to the transcript in either tool, click on **Show Event Report** (to just see the events), and then take a look at **G2S_NAE114** – the **Note Stacked** event. This is the event that is sent as soon as the note meters are updated, and it is used to convey this data model meter update to the host.

- Back in the RGS **Data Model Viewer**, select the Snapshot Icon again, to create another data model snapshot for this EGM, this time naming it “**snap2**”. Once the operation is complete, select the **Compare Data Models** tab and we’ll take a look at the comparison feature.

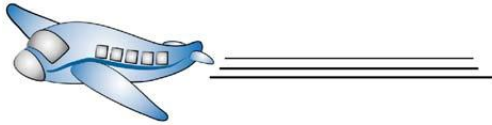


11. On the **Compare Data Models** tab, we see a listing of all of the data model snapshots that are being stored by the tool. Since there should only be two snapshots, you can press the **Select All** icon to select both of them (alternately, Ctrl-Left Click allows you to select specific snapshots). Once two snapshots are selected, press the **Compare** button to have the RGS compare the two instances of the data model and report any differences between the older and newer versions.



12. When you compare two snapshots, the information that changed is displayed in a new Compare Snapshot Deltas tab and is displayed by information type (meter, status, profile, and then option changes). You can easily filter the displayed data to see just the changed information that is important, then you can add comments to selected rows (just double-click the any row) and print the current view to share with others.

Flying Solo Activity 6-1



There are several ways to filter the comparison data in the Snapshot Comparison screen - by using one of the several Filter buttons (to apply a general filter that reduces the list), or by using the Quick Filter (our old friend from the Transcript exercises). Here are some tasks to familiarize you with each filter.

Task 1 Go back to the RST and generate a bit more activity – use the following as a guide:

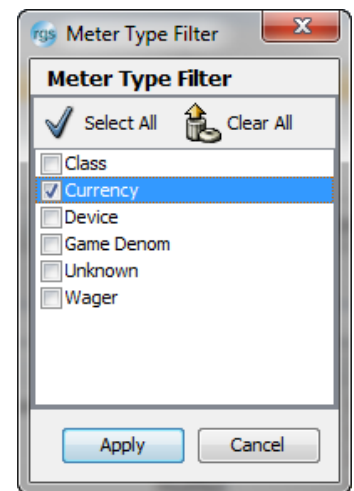
- Insert a \$10 note, having it go to the dispenser
- Play a Simple Game (gamePlay device 1, wagering \$3 and winning \$5).
- Insert a \$20 note, having it go to the drop
- Play another Simple Game (gamePlay device 3, wagering \$10 and winning \$5)

Task 2 Back in the RGS, create a new snapshot (**snap3**), and the compare **snap1** to **snap3**.

Task 3 Select the Meter Deltas Tab. Use **Filter by Meter Type** to select only the Currency meters. When you apply this filter, you will notice that the gamePlay activity is filtered out as only *currency* devices (such as the noteAcceptor) have *currency* meters. Also, you can see that the *currency* meters provide a total by each denomination of note.

Task 4 Now, use the following **Filter by Meter Type** options to see how each affects your view of the comparison information:

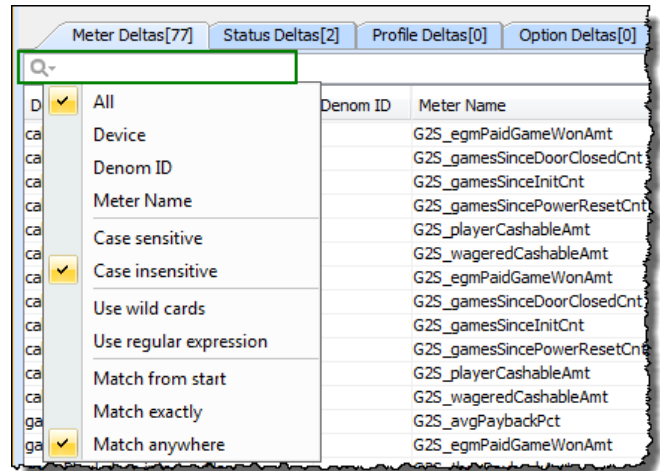
- **Select All** - Click to select all meter types. All meter delta records are included in the comparison.
- **Clear All** - Click to deselect all meter types. At this point, nothing should be in the comparison.
- **Class** – The Class meters provide the summarized totals for of all devices in the class. For noteAcceptors, there is typically just one device in the class – but there are 6 gamePlay devices in the RST. Try playing different games and then compare the **Class** meters to the **Device** meters.



- **Device** – In G2S, each device has its own set of meters, so you can see the performance of each gamePlay device.
- **Game Denom** – Like the **Currency** meters, the gameDenom meters report wagers at each bet denomination for each gamePlay device (also for the EGM).
- **Wager** – Since a payable can be divided into multiple wager categories, G2S provides a meter reporting mechanism by which an EGM can report wagers each of these different wager categories.

Task 5 Quick Filter – Back in the **Filter by Meter Type** control, click on **Select All**, so all meters deltas are again included in the report. If you click on the magnifying

glass at the left end of the Quick Filter you'll see that this instance of the Quick Filter allows you to search the **Device, Denom ID, and Meter Name** columns for the entered string (using Release 42 of the tools). By default, all are selected, but the search can be limited if desired:

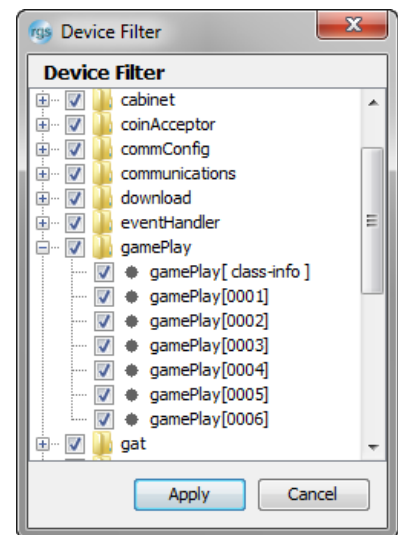


Use the **Quick Filter** (just below the Meter Deltas tab) to search for “cnt” to see all of the Count meters, for “pct” to see all of the percentage meters, and “10.” to see all of the \$10 currency meters.

As you can see – this is a really powerful tool!

Task 6 Device Filter – another filter that is handy for working with the DMV comparison report is the Device Filter (**Filter by Device**). As you can see, you can use this filter to easily select or deselect device classes, class level information, or even individual device-level information.

Try this filter out by selecting and deselecting information and then reviewing the results in the Comparison table.

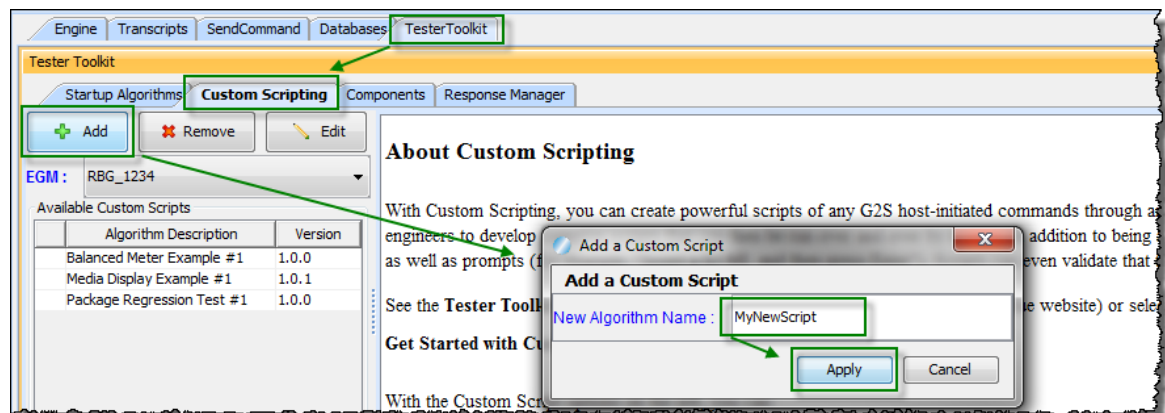


Activity 6-5: Scripting the Host Engine (RGS)

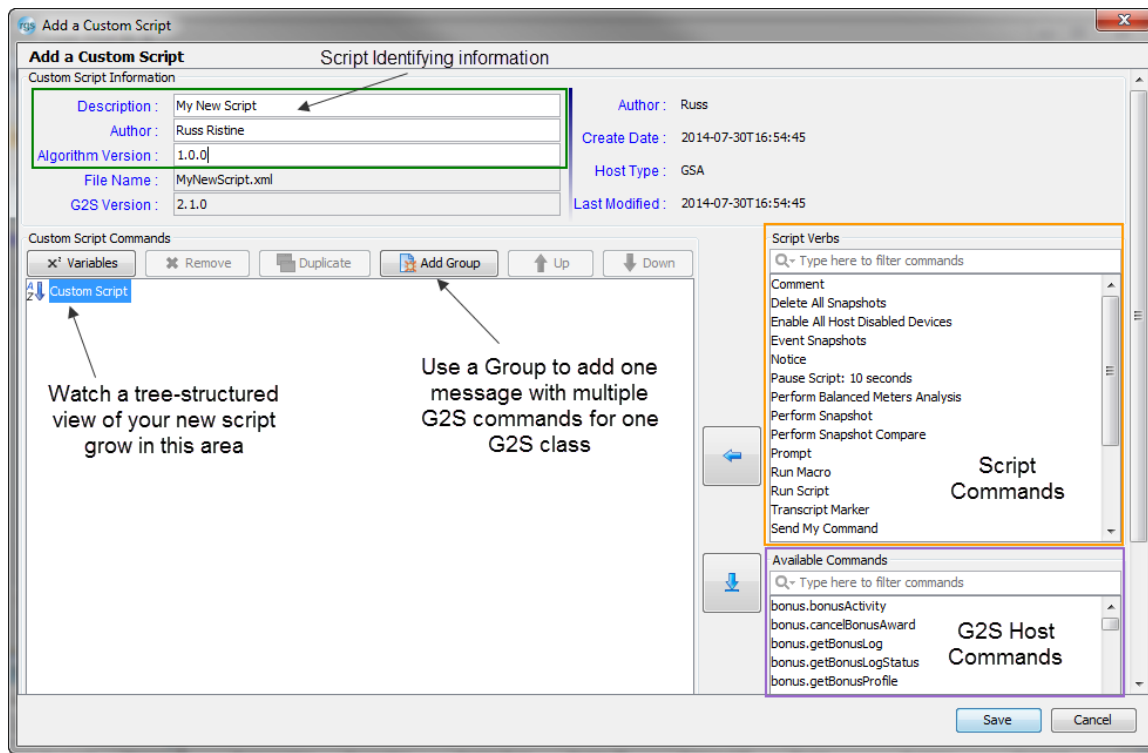
So far, the RGS has shown itself to be quite capable, as you can send any individual command to the EGM, and then examine the results in the Transcript, possibly using the Advanced Transcript Analyzer to quickly review large data sets. However, if you are doing repeated tests, it would be nice to automate sequences of G2S commands in a script, and then be able to run that script over and over again. The Custom Scripting engine in the RGS Tester Toolkit satisfies this requirement.

In addition to supporting nearly every G2S host initiated command, the Custom Scripting engine contains a set of Script Verbs that are used to augment the G2S functions by automating some of the functions of the RGS, itself. In this Activity, we'll create a basic script that requests the device profiles from the devices in several classes. In the next Activity, we'll try out some of the more advanced scripting functions.



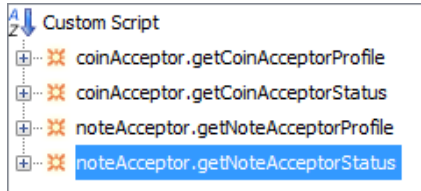
1. Start the RST and RGS if they are not already running. Start the SmartEGM in the RST so it's ready to receive G2S commands from the host.
2. In the RGS, select the TesterToolkit layout, and then the Custom Scripting tab.

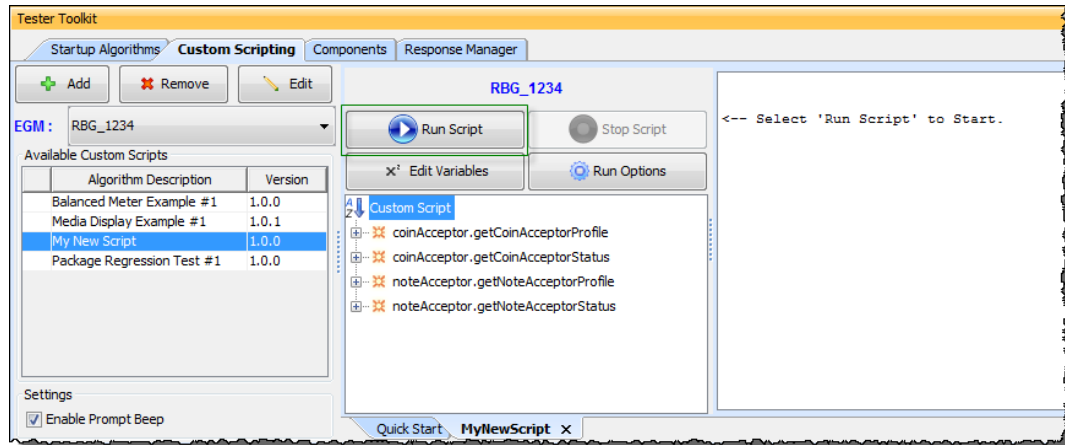


- a. Press the **Add** button to launch the **Add a Custom Script** window
- b. Enter a filename for your script (MyNewScript) and then press **Apply** to create the new script and launch the Custom Script editor.



3. In the Custom Script editor, you can modify the Script Identifying Information in the top portion of the screen, and then start building your script. The bottom portion of the control is divided into three parts:
 - a. **Custom Script Command** window, which contains a tree structured view of your script – making it very easy to review and modify any of the commands in your script.

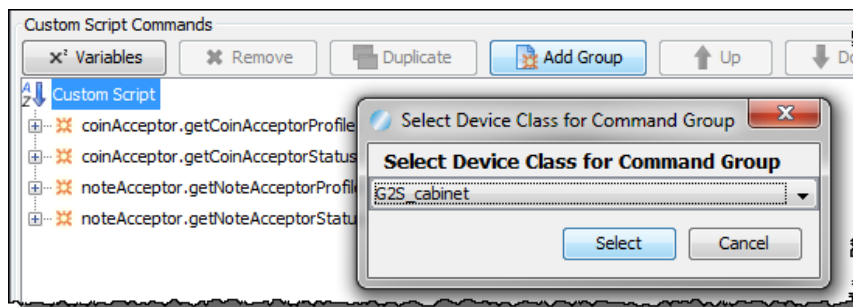
- b. **Script Verb** window, containing a list of the RGS specific commands that can be executed within a script. These script verbs allow you to work with data model snapshots, interact with the person running the script, and even run other scripts and macros from within a script.
 - c. **Available Commands** window provides a listing of the G2S Host Commands that are currently supported by the scripting engine. You'll notice that there's a Quick Filter panel at the top of this list to make it easy to find a specific set of commands in the list.
 4. Let's start with adding a few get commands for the coinAcceptor and noteAcceptor devices to our script. In the G2S Host Commands section, first type **coin** in the quick filter panel to reduce the list of G2S commands to those containing "coin" (which you'll see is the coinAcceptor commands. Select the `getCoinAcceptorProfile` command, and then Ctrl-Click on the `getCoinAcceptorStatus` command so both are selected in the Available Command list. Now press the  button in the middle of the screen to add these commands to the script tree.
 5. Next, type **notea** in the quick filter panel to filter the list to the noteAcceptor commands. Select the `getNoteAcceptorProfile` and `getNoteAcceptorStatus` commands, and then again press the  button to add them to the bottom of the script.
 6. At this point, your Custom Script "tree" should look something like the picture on the right, with two coinAcceptor commands followed by two noteAcceptor commands.
 7. Save your command script, using the **Save** button on the bottom right of the control, which closes the control.
 8. You'll notice that "My New Script" has been added to the list of **Available Command Scripts**. If you select it, you'll see that it opens in the panel on the right side of the Custom Scripting object.




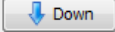
9. Clear the Transcript object in the bottom half of the layout (press **Clear > Display**), then **Run** your script. In the transcript, you'll see
 - a. a *Transcript Marker* record is added to the transcript indicating the start of the script “My New Script”,
 - b. 4 G2S commands are then shown, with responses from the EGM. Each command is identified as being generated by the Custom Scripting Engine, via a script called “My New Script”.
 - c. Finally, a second marker is added to indicate the end of the script.
 - d. If you want to run the script again, just press the **Run Script** button again.

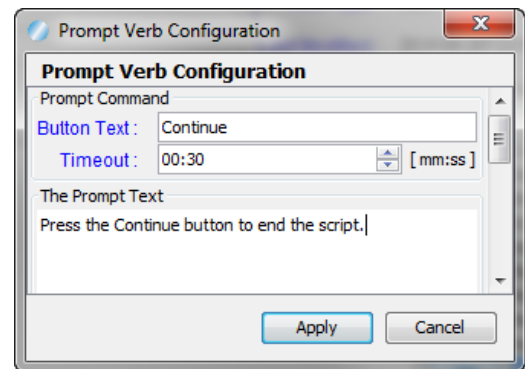
In G2S, you can also send multiple G2S commands in one message (up until now all commands have been in their own message). Let's now add a “Group” message to our Custom Script, as follows:

1. Select “My New Script” in the list of **Available Custom Scripts**, and then press the **Edit** button (or you can just double-click the script) to launch the Custom Script Editor window.



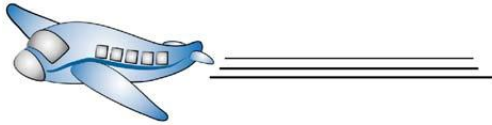
2. Make sure the “Custom Script” node at the top of the tree is high-lighted (as shown above – select it if it's not), then press the Add Group button to launch the Command

- Group dialogue box. For our example, select **G2S_cabinet** from the drop-down list then press the **Select** button, which adds a group to the bottom of the tree, and highlight the new G2S_cabinet Command Group node.
- In the G2S Host Commands section, type **cabinet** in the Quick Filter window to limit the list to cabinet commands. Select all 6 of the “get” commands (click on the first, and then Shift-Click on the last to easily select a group), then press the  button to add all 6 of the cabinet get commands to the cabinet Command Group.
 - Close the Command Group and highlight the command above the command group (Script Verbs are not allowed to be part of a Command Group so we need to get a little tricky here). Now select the **Prompt** verb and add some text to indicate that the script is over, as shown in the picture to the right. Press **Apply** to add the prompt to the script.
 - In the script tree, the new **Prompt** verb is highlighted, and is located just before the Command Group verb. Use the  button to move the **Prompt** verb to the bottom of the script, and then save the script to save your changes. Notice that your changes are also automatically applied to the active copy of the script.
 - Clear the transcript and run the script again. Notice that the **Prompt** verb has a 30 second timeout. If you don't press the continue button, the script automatically continues when the timeout expires.
 - In the Transcript, you'll see that the RGS has automatically separated each of the 6 cabinet commands, recording one per row in the transcript table. However, you can see that they were actually all sent in one message as there's a single g2sAck for all 6 commands in the group.



At this point, take another short break, and then we'll finish up with the Balanced Meters Report script...

Flying Solo Activity 5-1: The Balanced Meters Report (RGS)



During our study of G2S, we were fascinated by the balancing rules for meters that can be found in Appendix B of the G2S protocol document. Since the rules were well documented, we figured that we should see if we could make a verb for the Custom Scripting engine that would compare two snapshots, and then apply the rules from Appendix B to all of the meter changes between the two snapshots to ensure that the meter updates were done correctly by the EGM. By having this verb in a script, it makes it quite easy to check the EGM's meter balancing model.

1. As a final activity for this guide, select the **Balanced Meter Example #1** custom script that ships with the product. [Note: If you have an older version of the RGS (pre-42), the script was originally called "First GSA Script" – which is basically the same script.]
2. This script does the following:
 - a. Creates a snapshot called "Snap 1"
 - b. Prompts the tester (you) to do some activities on the EGM (in our case the RST), giving you 5 minutes to get it done. Done sooner, just press **Continue**.
 - c. Creates a second snapshot called "Snap 2"
 - d. Performs a Balance Meters Analysis (using the script verb of the same name) to analyze the meter changes between the two snapshots.
 - e. Displays the results so they can be easily reviewed.
3. Run the script and try it out.
4. Once the Balanced Meters report appears, try the following:
 - a. Double click on the tab at the bottom of the active script window (probably labelled **balanced-meter-example-001**). This should cause the active script window to undock from the RGS so you can see more of the comparison at once. To redock the control, just close the undocked window.
 - b. If you expand any of the + symbols, it will show all of the meters that made up that calculation.

- c. You can also Export this report to Excel – either the entire report, or just the errors. At this point, comparisons that weren't done are reported as errors – which will be fixed in a future release of the RGS. Stay tuned...

Name		Value 1	Value 2
EGM Balance in Total	✓	Debit: 22.00	Credit: 22.00
Debit			
noteAcceptor[class-level].G2S_currencyInAmt		16.00	
cabinet[class-level].G2S_egmPaidGameWonAmt		6.00	
Credit			
voucher[class-level].G2S_cashableOutAmt		19.75	
cabinet[class-level].G2S_wageredCashableAmt		9.00	
cabinet[class-level].G2S_playerCashableAmt		-6.75	
EGM Balance by Credit Type			
Cashable Credits	✓	Debit: 22.00	Credit: 22.00
Debit			
noteAcceptor[class-level].G2S_currencyInAmt		16.00	
cabinet[class-level].G2S_egmPaidGameWonAmt		6.00	
Credit			
voucher[class-level].G2S_cashableOutAmt		19.75	
cabinet[class-level].G2S_wageredCashableAmt		9.00	
cabinet[class-level].G2S_playerCashableAmt		-6.75	
Promo Credits	✓	Debit: 0.00	Credit: 0.00
Non-Cashable Credits	✓	Debit: 0.00	Credit: 0.00

Conclusion - The wrap-up

Hopefully you have found this Quick Start Guide to be a useful guide to the RadBlue tools, while providing some insight into GSA's Game to System (G2S) protocol. If you have any ideas for improving this guide, our tools, or have any questions on any of this stuff, just let us know. Our e-mail (support@radblue.com) is always open.

Good luck, and have fun with G2S!